IBM DB2 10.1
for Linux, UNIX and Windows

Datenbankverwaltung - Konzepte und Konfiguration - Referenzinformationen

SC12-4673-00

IBM DB2 10.1
for Linux, UNIX and Windows

Datenbankverwaltung - Konzepte und Konfiguration - Referenzinformationen

SC12-4673-00

Hinweis

Vor Verwendung dieser Informationen und des darin beschriebenen Produkts sollten die allgemeinen Informationen in Anhang B, „Bemerkungen", auf Seite 1003 gelesen werden.

Diese Veröffentlichung ist eine Übersetzung des Handbuchs
IBM DB2 10.1 for Linux, UNIX, and Windows, Database Administration Concepts and Configuration Reference,
IBM Form SC27-3871-00,
herausgegeben von International Business Machines Corporation, USA

© Copyright International Business Machines Corporation 1993, 2012

Informationen, die nur für bestimmte Länder Gültigkeit haben und für Deutschland, Österreich und die Schweiz nicht zutreffen, wurden in dieser Veröffentlichung im Originaltext übernommen.

Möglicherweise sind nicht alle in dieser Übersetzung aufgeführten Produkte in Deutschland angekündigt und verfügbar; vor Entscheidungen empfiehlt sich der Kontakt mit der zuständigen IBM Geschäftsstelle.

Änderung des Textes bleibt vorbehalten.

Herausgegeben von:
TSC Germany
Kst. 2877
Mai 2012

Inhaltsverzeichnis

Zu diesem Handbuch

Das Handbuch *Datenbankverwaltung - Konzepte und Konfiguration - Referenzinformationen* enthält Informationen zur Planung und zum Entwurf von Datenbanken sowie zur Implementierung und Verwaltung von Datenbankobjekten. Darüber hinaus enthält dieses Handbuch Referenzinformationen zur Konfiguration und Optimierung von Datenbanken.

Zielgruppe

Dieses Handbuch ist in erster Linie für Datenbank- und Systemadministratoren vorgesehen, die eine Datenbank für den Zugriff durch lokale und ferne Clients entwerfen, implementieren und verwalten müssen. Es bietet sich auch für Programmierer und andere Benutzer an, die sich Kenntnisse über die Verwaltung und den Betrieb des DB2-Verwaltungssystems für relationale Datenbanken aneignen müssen.

Aufbau des Handbuchs

Dieses Handbuch ist in vier Teile untergliedert. Die Teile 1 bis 3 bieten eine Übersicht über die Konzepte des DB2-Datenbankprodukts. Den Anfang bilden allgemeine Konzepte von Datenservern. Daran anschließend werden die verschiedenen Objekte behandelt, aus denen sich DB2-Datenbanken zusammensetzen. Teil 4 enthält Referenzinformationen.

Teil 1. Datenserver
Dieser Teil enthält eine kurze Beschreibung von DB2-Datenservern mit Informationen zur Verwaltung ihrer Kapazität sowie zur Unterstützung großer Seiten in 64-Bit-Umgebungen unter AIX. Darüber hinaus enthält er Informationen zur Ausführung mehrerer DB2-Kopien auf einem einzigen Computer, Informationen zu den automatischen Funktionen, die Sie bei der Verwaltung des Datenbanksystems unterstützten, Informationen zum Entwerfen, Erstellen und Verwenden von Instanzen sowie optionalen Informationen zur Konfiguration von LDAP-Servern (LDAP - Lightweight Directory Access Protocol).

Teil 2. Datenbanken
Dieser Teil beschreibt das Entwerfen, Erstellen und Verwalten von Datenbanken, Pufferpools, Tabellenbereichen und Schemata. Detaillierte Informationen zu Datenbankpartitionen finden Sie im Handbuch *Partitionierung und Clustering*.

Teil 3. Datenbankobjekte
Dieser Teil beschreibt das Entwerfen, Erstellen und Verwalten der folgenden Datenbankobjekte: Tabellen, Integritätsbedingungen, Indizes, Trigger, Sequenzen und Sichten.

Teil 4. Referenz
Dieser Teil enthält Referenzinformationen zur Konfiguration und Optimierung des Datenbanksystems durch Umgebungsvariablen, Registrierdatenbankvariablen und Konfigurationsparameter. Darüber hinaus enthält er Informationen zu den verschiedenen Namenskonventionen sowie zu SQL- und XML-Begrenzungen.

Teil 1. Datenserver

Kapitel 1. DB2-Datenserver

Datenserver stellen Software-Services für ein sicheres und effizientes Management für strukturierte Daten bereit. DB2 ist Hybridtyp aus relationalem Datenserver und XML-Datenserver.

Die Bezeichnung Datenserver bezieht sich auf einen Computer, auf dem die DB2-Datenbanksteuerkomponente installiert ist. Die DB2-Steuerkomponente ist ein umfassend ausgestattetes, leistungsfähiges Datenbankverwaltungssystem, das eine für die tatsächliche Datenbanknutzung optimierte SQL-Unterstützung sowie Tools zur Verwaltung der Daten enthält.

IBM® bietet eine Reihe von Datenserverprodukten an, zu denen auch Datenserver-Clients gehören, die auf sämtliche verschiedene Datenserver zugreifen können. Eine vollständige Liste der DB2-Datenserverprodukte und der verfügbaren Zusatzeinrichtungen sowie detaillierte Beschreibungen und Spezifikationen finden Sie unter http://www.ibm.com/software/data/db2/9/.

Verwaltung der Datenserverkapazität

Wenn die Kapazität des Datenservers nicht Ihren gegenwärtigen oder zukünftigen Anforderungen genügt, können Sie die Kapazität des Datenservers erweitern, indem Sie Plattenspeicherplatz hinzufügen und weitere Container erstellen oder Hauptspeicher hinzufügen. Wenn diese einfachen Strategien nicht die erforderliche Kapazität erbringen, können Sie auch das Hinzufügen von Prozessoren oder physischen Partitionen in Betracht ziehen.

Wenn Sie Ihr System durch eine Änderung der Umgebung skalieren, sollten Sie sich über die Auswirkungen dieser Änderung auf die Prozeduren in Ihrer Datenbank wie zum Beispiel auf das Laden von Daten oder das Backup oder den Restore von Datenbanken im Klaren sein.

Hinzufügen von Prozessoren

Wenn eine Konfiguration mit Einzelpartitionsdatenbanken und einem Einzelprozessor bis zur maximalen Kapazitätsgrenze ausgelastet wird, können Sie entweder Prozessoren oder logische Partitionen hinzufügen. Der Vorteil hinzugefügter Prozessoren liegt in der größeren Verarbeitungskapazität. Bei einer Konfiguration mit einer Einzelpartitionsdatenbank mit mehreren Prozessoren (SMP) werden Speicher und Speichersystemressourcen von den Prozessoren gemeinsam genutzt. Alle Prozessoren befinden sich in einem einzigen System, sodass die Auslastung nicht durch Kommunikation oder die Koordination von Tasks die Workload erhöht wird. Dienstprogramme, z. B. Programme zum Laden, Backup und Restore, können die zusätzlichen Prozessoren vorteilhaft nutzen.

Anmerkung: Einige Betriebssysteme, wie zum Beispiel das Solaris-Betriebssystem, können Prozessoren dynamisch in den Online- und den Offlinemodus versetzen.

Wenn Sie Prozessoren hinzufügen, prüfen und modifizieren Sie einige Datenbankkonfigurationsparameter, die die Anzahl der verwendeten Prozessoren bestimmen. Die folgenden Datenbankkonfigurationsparameter bestimmen die Anzahl der verwendeten Prozessoren und müssen möglicherweise aktualisiert werden:

- Standardgrad (**dft_degree**)

- Maximaler Grad der Parallelität bei Abfragen (**max_querydegree**)
- Partitionsinterne Parallelität aktivieren (**intra_parallel**)

Darüber hinaus sollten Sie auch Parameter prüfen, die bestimmen, wie Anwendungen die parallele Verarbeitung ausführen.

In einer Umgebung, in der TCP/IP zur Kommunikation verwendet wird, prüfen Sie den Wert der Registrierdatenbankvariablen **DB2TCPCONNMGRS**.

Hinzufügen zusätzlicher Computer

Wenn Sie eine vorhandene Umgebung mit partitionierten Datenbanken haben, können Sie die Verarbeitungskapazität und die Datenspeicherkapazität erhöhen, indem Sie der Umgebung zusätzliche Computer (entweder jeweils mit einem oder mit mehreren Prozessoren) sowie Speicherressourcen hinzufügen. Die Hauptspeicher- und Massenspeicherressourcen werden unter den Computern nicht gemeinsam genutzt. Diese Option hat den Vorteil, dass Daten und Benutzerzugriffe auf Speichereinheiten und Computer verteilt werden.

Nach dem Hinzufügen der neuen Computer und Speichereinheiten verwenden Sie den Befehl **START DATABASE MANAGER**, um den neuen Computern neue Datenbankpartitionsserver hinzuzufügen. Für jede Datenbank in der Instanz auf jedem neuen Datenbankpartitionsserver, den Sie hinzufügen, wird eine neue Datenbankpartition erstellt und konfiguriert. In den meisten Fällen brauchen Sie die Instanz nach dem Hinzufügen der neuen Datenbankpartitionsserver nicht erneut zu starten.

Aktivieren der Unterstützung großer Seiten (AIX)

Zum Aktivieren der Unterstützung großer Seiten in DB2-Datenbanksystemen unter AIX-Betriebssystemen müssen Sie einige Betriebssystemparameter konfigurieren und anschließend die Registrierdatenbankvariable **DB2_LARGE_PAGE_MEM** definieren.

Vorbereitende Schritte

Sie müssen über die Rootberechtigung verfügen, um mit AIX-Betriebssystembefehlen arbeiten zu können.

Informationen zu diesem Vorgang

Zusätzlich zur traditionellen Seitengröße von 4 KB unterstützen die POWER4-Prozessoren (und höhere Versionen) unter System z auch eine Seitengröße von 16 MB. Anwendungen, wie zum Beispiel IBM DB2 Version 10.1 for AIX, die intensiven Zugriff auf den Hauptspeicher erfordern und große Mengen an virtuellem Speicher nutzen, können durch die Verwendung großer Seiten eine Leistungsverbesserung erzielen.

Anmerkung:

1. Die Einstellung der Registrierdatenbankvariablen **DB2_LARGE_PAGE_MEM** impliziert ebenfalls, dass der Speicher fixiert wird.
2. Sie müssen äußerst vorsichtig vorgehen, wenn Sie Ihr System für das Fixieren (Pinning) von Speicher und die Unterstützung großer Seiten konfigurieren. Wenn Sie zu viel Speicher fixieren, führt das zu aufwendigen Seitenwechselvorgängen für Hauptspeicherseiten, die nicht im Speicher belassen werden. Wenn Sie zu viel physischen Speicher für große Seiten zuordnen, wird die Systemleistung beeinträchtigt, wenn nicht ausreichend Speicher für die Unterstützung der 4-KB-Seiten vorhanden ist.

Einschränkungen

Eine Aktivierung der Unterstützung großer Seiten verhindert, dass der Speichermanager für die automatische Leistungsoptimierung die Datenbankspeicherbelegung automatisch optimiert. Daher sollte sie nur für klar definierte Auslastungen, die durch einen relativ statischen Datenbankspeicherbedarf gekennzeichnet sind, in Betracht gezogen werden.

Vorgehensweise

Gehen Sie wie folgt vor, um die Unterstützung großer Seiten in DB2-Datenbanksystemen unter AIX-Betriebssystemen zu aktivieren:

1. Konfigurieren Sie Ihren AIX-Server für die Unterstützung großer Seiten, indem Sie den Befehl **vmo** mit den folgenden Markierungen absetzen:

   ```
   vmo -r -o lgpg_size=GroßeSeitengröße -o lgpg_regions=GroßeSeiten
   ```

   Dabei gibt *GroßeSeitengröße* die Größe der hardwareunterstützten großen Seiten in Byte und *GroßeSeiten* die Anzahl der zu reservierenden großen Seiten an. Wenn Sie zum Beispiel 25 GB für die Unterstützung großer Seiten zuordnen möchten, führen Sie den Befehl folgendermaßen aus:

   ```
   vmo -r -o lgpg_size=16777216 -o lgpg_regions=1600
   ```

   Detaillierte Informationen zur Ausführung des Befehls **vmo** finden Sie in Ihren AIX-Handbüchern.
2. Führen Sie den Befehl **bosboot** aus, sodass der zuvor ausgeführte Befehl **vmo** nach dem nächsten Systemstart wirksam wird.
3. Aktivieren Sie den Server nach dessen Start für fixierten ('pinned') Speicher. Führen Sie den Befehl **vmo** mit den folgenden Markierungen aus:

   ```
   vmo -o v_pinshm=1
   ```
4. Setzen Sie mithilfe des Befehls **db2set** die Registrierdatenbankvariable **DB2_LARGE_PAGE_MEM** auf DB und starten Sie anschließend den DB2-Datenbankmanager. Beispiel:

   ```
   db2set DB2_LARGE_PAGE_MEM=DB
   db2start
   ```

Ergebnisse

Nach Ausführung der beschriebenen Schritte weist das DB2-Datenbanksystem das Betriebssystem zur Verwendung eines Speichers mit großen Seiten für den gemeinsamen Datenbankspeicherbereich an.

Fixieren von gemeinsamem DB2-Datenbankspeicher (AIX)

Zum Fixieren von gemeinsamem DB2-Datenbankspeicher unter AIX-Betriebssystemen müssen Sie einige Betriebssystemparameter konfigurieren und anschließend die Registrierdatenbankvariable **DB2_PINNED_BP** definieren.

Vorbereitende Schritte

Sie müssen über Rootberechtigung verfügen, um die AIX-Betriebssystembefehle ausführen zu können.

Informationen zu diesem Vorgang

Der Vorteil beim Fixieren von Teilen von Speicher liegt darin, dass beim Zugriff auf eine fixierte Seite diese Seite ohne Verwendung des Algorithmus zum Ersetzen von Seiten abgerufen werden kann. Ein Nachteil besteht darin, dass Sie darauf achten müssen, dass das System nicht überlastet wird, da die Flexibilität des Betriebssystems hinsichtlich der Speicherverwaltung reduziert wird. Wenn Sie zu viel Speicher fixieren, führt das zu aufwendigen Seitenwechselvorgängen für Hauptspeicherseiten, die nicht im Speicher belassen werden.

Einschränkungen

Wenn Sie die Registrierdatenbankvariable **DB2_PINNED_BP** auf YES setzen, kann die automatische Leistungsoptimierung für gemeinsamen Datenbankspeicher nicht aktiviert werden.

Vorgehensweise

Gehen Sie wie folgt vor, um gemeinsamen DB2-Datenbankspeicher unter AIX-Betriebssystemen zu fixieren:

1. Konfigurieren Sie das Betriebssystem AIX entsprechend, um fixierten Speicher zu aktivieren. Führen Sie den Befehl **vmo** mit den folgenden Markierungen aus:

   ```
   vmo -o v_pinshm=1
   ```

   Detaillierte Informationen zur Ausführung des Befehls **vmo** finden Sie in Ihren AIX-Handbüchern.
2. (Optional) Wenn Sie mittlere Seitengrößen verwenden (die Standardfunktionsweise), müssen Sie sicherstellen, dass der DB2-Instanzeigner über die Funktionen CAP_BYPASS_RAC_VMM und CAP_PROPAGATE verfügt. Beispiel:

   ```
   chuser capabilities=CAP_BYPASS_RAC_VMM,CAP_PROPAGATE db2inst1
   ```

   Dabei ist *db2inst1* die Benutzer-ID des DB2-Instanzeigners.
3. Führen Sie den Befehl **bosboot** aus, damit der Befehl **vmo** nach dem nächsten Systemstart wirksam wird.
4. Aktivieren Sie nach dem Starten des Servers das DB2-Datenbanksystem für fixierten ('pinned') Speicher.
   a. Setzen Sie den Befehl **db2set** ab, um die Registrierdatenbankvariable **DB2_PINNED_BP** auf YES zu setzen.
   b. Starten Sie den DB2-Datenbankmanager.

   Beispiel:

   ```
   db2set DB2_PINNED_BP=YES
   db2start
   ```

Ergebnisse

Nach Ausführung der beschriebenen Schritte weist das DB2-Datenbanksystem das Betriebssystem zum Fixieren des gemeinsamen DB2-Datenbankspeichers an.

Kapitel 2. Mehrere DB2-Kopien - Übersicht

Mit Version 9 oder einer späteren Version können Sie mehrere DB2-Kopien auf demselben Computer installieren und ausführen. Eine DB2-Kopie bezieht sich auf eine oder auch mehrere Installationen von DB2-Datenbankprodukten an einer bestimmten Position auf demselben Computer. Die Kopien von DB2 Version 9 können dieselbe oder auch verschiedene Codeversionen besitzen.

Dies bietet folgende Vorteile:

- Die Möglichkeit, Anwendungen auszuführen, die verschiedene DB2-Datenbankversionen auf demselben Computer zur gleichen Zeit erfordern.
- Die Möglichkeit, unabhängige Kopien von DB2-Datenbankprodukten für unterschiedliche Funktionen auszuführen.
- Die Möglichkeit, auf demselben Computer Tests auszuführen, bevor eine Produktionsdatenbank auf eine neuere Version des DB2-Datenbankprodukts migriert wird.
- Die Möglichkeit für unabhängige Softwareanbieter (ISVs), ein DB2-Datenbankserverprodukt in ihr Produkt einzubetten und die DB2-Datenbank den Benutzern gegenüber zu verdecken. Verwenden und verteilen Sie für COM+-Anwendungen den IBM Data Server Driver for ODBC and CLI mit Ihrer Anwendung und nicht den Data Server Runtime Client, da für COM+-Anwendungen jeweils nur ein Data Server Runtime Client verwendet werden kann. Der IBM Data Server Driver for ODBC and CLI unterliegt dieser Einschränkung nicht.

In Tabelle 1 sind die relevanten Themen in den einzelnen Kategorien aufgeführt.

Tabelle 1. Übersicht über die Informationen zu mehreren DB2-Kopien

Kategorie	Zugehörige Themen
Allgemeine Informationen und Einschränkungen	• „Standardkopie der IBM-Datenbankclientschnittstelle" auf Seite 8
Installation	• „Installieren von DB2-Servern (Linux und UNIX)" in *DB2-Server - Installation* • „Installieren von DB2-Servern (Windows)" in *DB2-Server - Installation*
Konfiguration	• „Einrichten des DB2-Verwaltungsservers bei Verwendung mehrerer DB2-Kopien" auf Seite 11 • „Festlegen der Standardinstanz bei Verwendung mehrerer DB2-Kopien (Windows)" auf Seite 12 • „dasupdt - Verwaltungsserver aktualisieren (Befehl)" in *Command Reference*
Verwaltung	• „Aktualisieren von DB2-Kopien (Windows)" auf Seite 17 • „Aktualisieren von DB2-Kopien (Linux und UNIX)" auf Seite 15 • „db2iupdt - Instanzen aktualisieren (Befehl)" in *Command Reference* • „db2swtch - DB2-Standardkopie wechseln (Befehl)" in *Command Reference* • „db2SelectDB2Copy (API) - DB2-Kopie zur Verwendung durch eine Anwendung auswählen" in *Administrative API Reference*

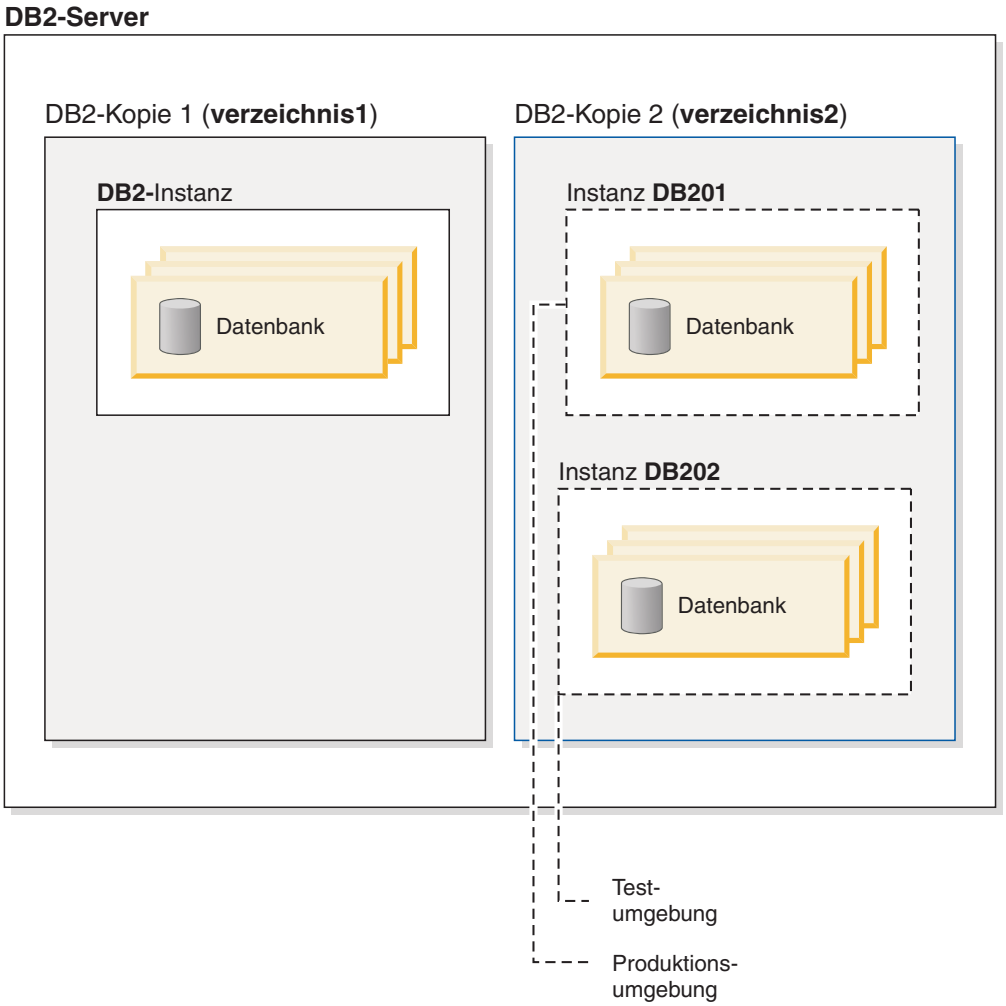
Standardkopie der IBM-Datenbankclientschnittstelle

Sie können über mehrere DB2-Kopien auf einem einzigen Computer sowie über eine Standardkopie der IBM Datenbankclientschnittstelle verfügen. Über diese Schnittstelle erhält eine Clientanwendung den ODBC-Treibercode, den CLI-Treibercode und den .NET-Datenprovidercode, der standardmäßig für eine Kommunikation mit der Datenbank erforderlich ist.

Seit Version 9.1 ist der Code für die Kopie der IBM Datenbankclientschnittstelle in der DB2-Kopie enthalten. Ab Version 9.5 steht ein neues Produkt zur Verfügung, das Sie installieren können und das den erforderlichen Code für die Kommunikation einer Clientanwendung mit einer Datenbank bereitstellt. Dieses Produkt ist ein IBM Data Server Driver Package (DSDRIVER). Mit Version 9.5 (und späteren Versionen) können Sie DSDRIVER auf einer IBM Data Server-Treiberkopie installieren, die von der Installation einer DB2-Kopie getrennt ist.

Ab Version 9.1 können mehrere DB2-Kopien auf Ihrem Computer installiert sein, ab Version 9.5 können mehrere IBM Datenbankclientschnittstellenkopien und mehrere DB2-Kopien auf Ihrem Computer installiert sein. Bei der Installation einer neuen DB2-Kopie oder einer neuen IBM Daten Server-Treiberkopie haben Sie die Möglichkeit, die DB2-Standardkopie und die Standardkopie der IBM Datenbankclientschnittstelle zu ändern.

Im folgenden Diagramm sind mehrere DB2-Kopien auf einem DB2-Server installiert. Dabei kann es sich um eine beliebige Kombination der DB2-Datenbankprodukte handeln:



Kopien der Version 8 und Version 9 (oder höher) können auf ein und demselben Computer koexistieren. Allerdings muss Version 8 als DB2-Standardkopie und als Standardkopie der IBM Datenbankclientschnittstelle verwendet werden. Es ist nicht möglich, während der Installation von der Kopie der Version 8 zur Kopie der Version 9 (oder höher) als DB2-Standardkopie oder als Standardkopie der IBM Datenbankclientschnittstelle zu wechseln. Es ist außerdem nicht möglich, später den Befehl zum Wechseln der Standardkopie **db2swtch** auszuführen, es sei denn, Sie führen zuerst ein Upgrade auf Version 9 (oder höher) durch oder deinstallieren die Kopie der Version 8. Wenn Sie den Befehl **db2swtch** ausführen, solange Version 8 noch auf dem System installiert ist, erhalten Sie eine Fehlernachricht, die Sie darüber informiert, dass die Standardkopie nicht geändert werden kann, weil auf dem System Version 8 gefunden wurde.

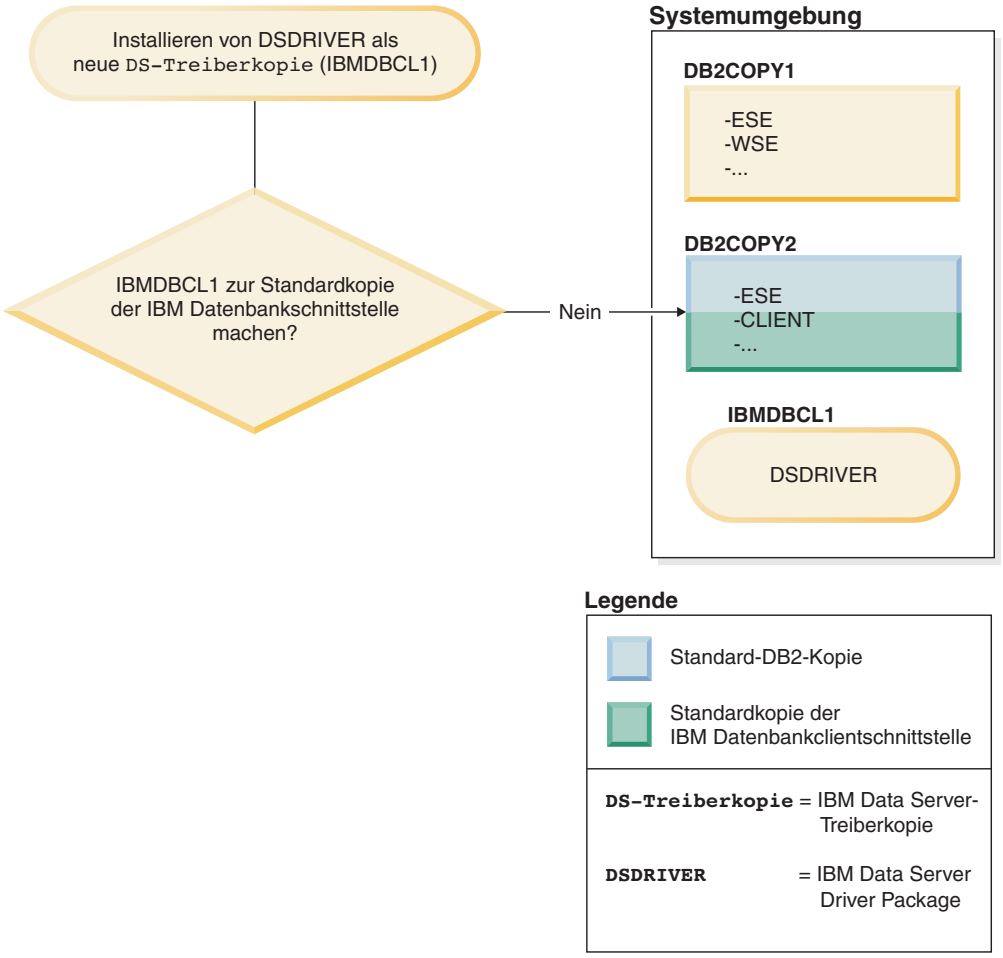
Nach der Installation mehrerer DB2-Kopien oder mehrerer IBM Datenservertreiberkopien können Sie die DB2-Standardkopie bzw. die Standardkopie der IBM Datenbankclientschnittstelle ändern, wenn Sie dies wünschen. Wenn Sie Version 8 installiert haben, müssen Sie das Produkt deinstallieren oder ein Upgrade des Produkts auf Version 9 (oder höher) durchführen, bevor Sie die DB2-Standardkopie oder die Standardkopie der IBM Datenbankclientschnittstelle ändern können.

Clientanwendungen können immer direkt an die Speicherposition des Datenservertreibers geleitet werden. Dies ist das Verzeichnis, in dem DSDRIVER installiert ist.

Wenn Sie die DB2-Kopie oder die IBM Daten Server-Treiberkopie, die die Standardkopie der IBM Datenbankclientschnittstelle ist, deinstallieren, werden die Standardkopien für Sie verwaltet. Ausgewählte Standardkopien werden entfernt und neue Standardkopien werden für Sie ausgewählt. Wenn Sie die DB2-Standardkopie deinstallieren, die jedoch nicht die letzte DB2-Kopie auf dem System ist, werden Sie aufgefordert, zunächst eine andere DB2-Kopie als Standardkopie festzulegen.

Auswählen einer Standardkopie beim Installieren einer neuen IBM Datenbankclientschnittstellenkopie

Ab Version 9.5 kommt ein Szenario in Betracht, in dem Sie zwei DB2-Kopien (DB2COPY1 und DB2COPY2) installiert haben können. DB2COPY2 ist die DB2-Standardkopie und die Standardkopie der IBM Datenbankclientschnittstelle.

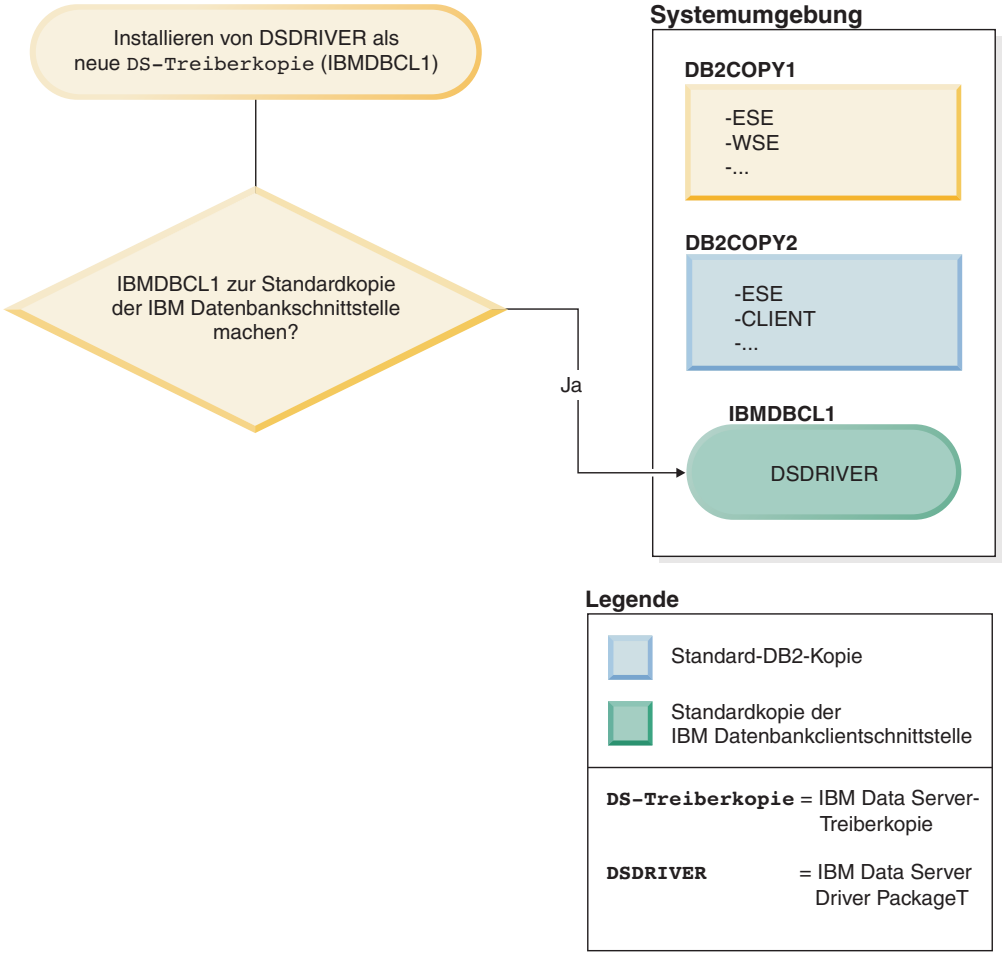


Sie installieren IBM Data Server Driver Package (DSDRIVER) auf einer neuen IBM Data Server-Treiberkopie.

Während der Installation der neuen IBM Data Server-Treiberkopie (IBMDBCL1) werden Sie gefragt, ob die neue IBM Data Server-Treiberkopie zur Standardkopie der IBM Datenbankclientschnittstelle gemacht werden soll.

Wenn Sie „Nein" antworten, wird DB2COPY2 als Standardkopie der IBM Datenbankclientschnittstelle beibehalten. (Außerdem fungiert diese Kopie weiter als DB2-Standardkopie.)

Betrachten Sie jedoch auch den Fall, in dem Sie im gleichen Szenario mit „Ja" auf die Frage antworten, ob die neue IBM Data Server-Treiberkopie zur Standardkopie der IBM Datenbankclientschnittstelle gemacht werden soll.



In diesem Fall wird IBMDBCL1 als Standardkopie der IBM Datenbankclientschnittstelle eingerichtet. (DB2COPY2 bleibt jedoch als DB2-Standardkopie erhalten.)

Einrichten des DB2-Verwaltungsservers bei Verwendung mehrerer DB2-Kopien

Seit Version 9.1 können Sie mehrere DB2-Kopien haben, die auf demselben Computer ausgeführt werden. Dies wirkt sich auf die Funktionsweise des DB2-Verwaltungsservers (DAS - DB2 Administration Server) aus.

Informationen zu diesem Vorgang

Der DAS ist eine einzigartige Komponente innerhalb des Datenbankmanagers, die auf eine aktive Version beschränkt ist, unabhängig davon, wie viele DB2-Kopien auf demselben Computer installiert sind. Aus diesem Grund gelten für ihn die folgenden Einschränkungen und funktionalen Anforderungen.

Einschränkungen

Wichtig: Der DB2-Verwaltungsserver (DAS) gilt in Version 9.7 als veraltet und wird möglicherweise in einem zukünftigen Release entfernt. Der DAS wird in DB2

pureScale-Umgebungen nicht unterstützt. Verwenden Sie Softwareprogramme, die das Secure Shell-Protokoll für die Fernverwaltung nutzen. Weitere Informationen hierzu finden Sie im Abschnitt „ DB2-Verwaltungsserver (DAS) gilt als veraltet" in http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0059276.html.

Auf dem Server darf nur eine DAS-Version installiert sein, die die Instanzen wie folgt verwaltet:

- Wenn der DAS unter Version 9.1 oder Version 9.5 ausgeführt wird, kann er zur Verwaltung von Instanzen der Version 8, der Version 9.1 oder Version 9.5 verwendet werden.
- Wenn der DAS unter Version 8 ausgeführt wird, kann er nur Instanzen der Version 8 verwalten. Sie können ein Upgrade für den DAS der Version 8 durchführen oder diese Version löschen und anschließend einen neuen DAS der Version 9.5 erstellen, um die Instanzen der Version 8 und späterer Versionen zu verwalten.

Ungeachtet der Anzahl der auf ein und demselben Computer installierten DB2-Kopien kann nur ein DAS auf diesem Computer zu gleicher Zeit erstellt werden. Dieser DAS wird von allen DB2-Kopien verwendet, die auf demselben Computer ausgeführt werden. In Version 9.1 oder einer späteren Version kann der DAS zu jeder beliebigen DB2-Kopie gehören, die momentan installiert ist.

Wenn der DAS in einer Kopie der Version 9.5 ausgeführt wird und in einer anderen Kopie der Version 9.5 ausgeführt werden soll, verwenden Sie den Befehl **dasupdt**. Wenn der DAS in einer Kopie der Version 8, Version 9.1 oder Version 9.5 ausgeführt wird und in einer Kopie der Version 9.7 ausgeführt werden soll, können Sie den Befehl **dasupdt** nicht verwenden. Verwenden Sie den Befehl **dasmigr**, um ein Upgrade des DAS auf Version 9.7 durchzuführen.

Unter Windows-Betriebssystemen können Sie den Befehl **dasupdt** auch verwenden, wenn Sie den DAS in einer neuen DB2-Standardkopie derselben Version ausführen müssen.

Vorgehensweise

Gehen Sie wie folgt vor, um den DAS in einer der DB2-Kopien einzurichten:

Wählen Sie eine der folgenden Aktionen aus:

- Wenn der DAS nicht erstellt wurde, erstellen Sie einen DAS in einer der DB2-Kopien.
- Verwenden Sie den Befehl **dasupdt** nur, um den DAS zu aktualisieren, sodass er in einer anderen DB2-Kopie desselben Release ausgeführt wird.
- Verwenden Sie den Befehl **dasmigr**, um ein Upgrade des DAS von Version 8, Version 9.1 oder Version 9.5 auf Version 9.7 durchzuführen.

Festlegen der Standardinstanz bei Verwendung mehrerer DB2-Kopien (Windows)

Die **DB2INSTANCE**-Umgebung wird entsprechend der DB2-Kopie definiert, auf deren Verwendung Ihre Umgebung momentan eingestellt ist. Wenn Sie sie nicht explizit auf eine Instanz in der aktuellen Kopie einstellen, wird die Standardinstanz verwendet, die durch die Variable **DB2INSTDEF** der Profilregistrierdatenbank angegeben ist.

Informationen zu diesem Vorgang

DB2INSTDEF ist die Standardinstanzvariable, die für die momentan verwendete DB2-Kopie spezifisch ist. Jede DB2-Kopie besitzt eine eigene Profilregistrierdatenbankvariable **DB2INSTDEF**. Instanznamen müssen auf dem System eindeutig sein. Wenn eine Instanz erstellt wird, durchsucht der Datenbankmanager die vorhandenen Kopien, um die Eindeutigkeit des Namens sicherzustellen.

Beachten Sie bei der Festlegung der Standardinstanz die folgenden Richtlinien, wenn Sie mit mehreren DB2-Kopien arbeiten:

- Wenn **DB2INSTANCE** für eine bestimmte DB2-Kopie nicht festgelegt ist, wird der Wert der Variablen **DB2INSTDEF** für diese DB2-Kopie verwendet. Dies bedeutet Folgendes:
  - Ist **DB2INSTANCE**=ABC und **DB2INSTDEF**=XYZ, wird der Wert ABC verwendet.
  - Ist **DB2INSTANCE** *nicht definiert* und **DB2INSTDEF**=XYZ, wird der Wert XYZ verwendet.
  - Ist **DB2INSTANCE** *nicht definiert* und **DB2INSTDEF** *nicht definiert*, können alle Anwendungen bzw. Befehle nicht ausgeführt werden, die von einem gültigen Wert für **DB2INSTANCE** abhängig sind.
- Sie können entweder den Befehl **db2envar.bat** oder die API db2SelectDB2Copy verwenden, um zwischen DB2-Kopien zu wechseln. Es ist auch möglich, alle Umgebungsvariablen (z. B. **PATH**, **INCLUDE**, **LIB** und **DB2INSTANCE**) richtig zu definieren, jedoch müssen Sie sicherstellen, dass sie ordnungsgemäß definiert werden.

  Anmerkung: Die Verwendung des Befehls **db2envar.bat** ist nicht ganz identisch mit der Einstellung der Umgebungsvariablen. Der Befehl **db2envar.bat** stellt fest, zu welcher DB2-Kopie er gehört, und fügt den Pfad dieser DB2-Kopie am Anfang der Umgebungsvariablen **PATH** an.

  Wenn mehrere DB2-Kopien auf demselben Computer vorhanden sind, kann die Umgebungsvariable **PATH** nur auf eine von ihnen verweisen: die Standardkopie (DEFAULT COPY). Angenommen z. B., die Standardkopie DB2COPY1 ist in c:\sqllib\bin gespeichert, während DB2COPY2 sich unter d:\sqllib\bin befindet. Wenn Sie DB2COPY2 in einem regulären Befehlsfenster verwenden wollten, würden Sie d:\sqllib\bin\db2envar.bat in diesem Befehlsfenster ausführen. Dadurch wird die Umgebungsvariable **PATH** (wie einige andere Umgebungsvariablen auch) für dieses Befehlsfenster angepasst, sodass die Binärdateien aus dem Verzeichnis d:\sqllib\bin verwendet werden.
- **DB2INSTANCE** ist nur für die Instanzen unter der DB2-Kopie gültig, die Sie verwenden. Wenn Sie jedoch die Kopie wechseln, indem Sie den Befehl **db2envar.bat** ausführen, wird **DB2INSTANCE** mit dem Wert von **DB2INSTDEF** für die DB2-Kopie aktualisiert, zu der Sie anfangs gewechselt haben.
- **DB2INSTANCE** gibt die aktuelle DB2-Instanz an, die von Anwendungen verwendet wird, die in dieser DB2-Kopie ausgeführt werden. Wenn Sie zwischen Kopien wechseln, wird **DB2INSTANCE** standardmäßig in den Wert von **DB2INSTDEF** für die betreffende Kopie geändert. **DB2INSTDEF** hat in einem System mit nur einer Kopie geringere Bedeutung, weil sich alle Instanzen in der aktuellen Kopie befinden. Jedoch ist sie auch in diesem Fall als Standardinstanz verwendbar, wenn keine andere Instanz definiert ist.
- Alle globalen Variablen der Profilregistrierdatenbank sind für eine DB2-Kopie spezifisch, sofern Sie sie nicht mit dem Befehl SET VARIABLE=*variablenname* angeben.

Mehrere Instanzen des Datenbankmanagers

Auf einem einzigen Server können mehrere Instanzen des Datenbankmanagers erstellt werden. Das heißt, dass verschiedene Instanzen desselben Produkts auf einem physischen Computer erstellt und gleichzeitig ausgeführt werden können. Dadurch haben Sie flexible Möglichkeiten bei der Einrichtung von Umgebungen.

Anmerkung: Ein Instanzname darf nicht in zwei unterschiedlichen DB2-Kopien verwendet werden.

Mehrere Instanzen können z. B. zum Erstellen der folgenden Umgebungen wünschenswert sein:

- Trennen der Entwicklungsumgebung von der Geschäftsumgebung.
- Getrenntes Optimieren jeder Umgebung für bestimmte Anwendungen, die ausgeführt werden sollen.
- Schützen sensibler Informationen vor Administratoren. Es kann zum Beispiel erforderlich sein, die Lohn- und Gehaltsdaten in einer eigenen Instanz geschützt zu speichern, sodass Eigner anderer Instanzen diese Daten nicht einsehen können.

Anmerkung:

- Nur für UNIX-Betriebssysteme: Stellen Sie zur Vermeidung von Umgebungskonflikten zwischen zwei oder mehr Instanzen sicher, dass sich die einzelnen Instanzausgangsverzeichnisse auf einem lokalen Dateisystem befinden.
- Nur für Windows-Betriebssysteme: Instanzen werden entweder als lokal (local) oder als fern (remote) im Knotenverzeichnis katalogisiert. Die Standardinstanz wird durch die Umgebungsvariable **DB2INSTANCE** definiert. Sie können die Verbindung zu anderen Instanzen (Befehl **ATTACH**) herstellen, um Wartungsaufgaben und Dienstprogramme auszuführen, die nur auf Instanzebene ausgeführt werden können, wie z. B. Erstellen einer Datenbank, Abbrechen von Anwendungen, Monitoraufzeichnungen für eine Datenbank oder Aktualisieren der Konfiguration des Datenbankmanagers. Wenn Sie versuchen, die Verbindung zu einer Instanz herzustellen, die nicht Ihre Standardinstanz ist, wird anhand des Knotenverzeichnisses festgestellt, wie die Kommunikation mit der anderen Instanz herzustellen ist.
- Auf allen Plattformen: DB2-Datenbankprogrammdateien werden physisch an einer Position gespeichert, und jede Instanz verweist wieder auf die Kopie, der diese Instanz angehört, sodass die Programmdateien bei der Erstellung der einzelnen erstellten Instanzen nicht kopiert werden. In einer Instanz können mehrere zusammengehörige Datenbanken gespeichert werden.

Mehrere Instanzen (Windows)

Es ist möglich, mehrere Instanzen des DB2-Datenbankmanagers auf demselben Computer auszuführen. Jede Instanz des Datenbankmanagers pflegt eigene Datenbanken und verfügt über eigene Konfigurationsparameter für den Datenbankmanager.

Anmerkung: Die Instanzen können auch unterschiedlichen DB2-Kopien auf einem Computer angehören, der unterschiedliche Versionen des Datenbankmanagers haben kann. Wenn Sie ein 64-Bit-Windows-System verwenden, können Sie eine 32-Bit-DB2-Version oder eine 64-Bit-DB2-Version installieren, diese können jedoch nicht auf derselben Maschine koexistieren.

Eine Instanz des Datenbankmanagers besteht aus folgenden Komponenten:

- Ein Windows-Dienst (Service), der die Instanz darstellt. Der Name des Dienstes ist mit dem Namen der Instanz identisch. Der Anzeigename des Dienstes (im Fenster **Dienste**) ist der Instanzname, dem die Zeichenfolge „DB2 - " vorangestellt ist. Zum Beispiel gibt es für eine Instanz mit dem Namen „DB2" einen Windows-Dienst mit dem Namen „DB2" der als „DB2 - *name_der_db2-kopie* - DB2" angezeigt wird.

  Anmerkung: Für Clientinstanzen wird kein Windows-Dienst erstellt.
- Ein Instanzverzeichnis. Dieses Verzeichnis enthält die Konfigurationsdateien des Datenbankmanagers, das Systemdatenbankverzeichnis, das Knotenverzeichnis, das DCS-Verzeichnis, alle Diagnoseprotokolldateien sowie alle Speicherauszugsdateien, die der Instanz zugeordnet sind. Das Instanzverzeichnis ist je nach Edition des Betriebssystems der Windows-Familie unterschiedlich. Zur Feststellung des Standardverzeichnisses unter Windows prüfen Sie die Einstellung der Umgebungsvariablen **DB2INSTPROF** mithilfe des Befehls **db2set DB2INSTPROF**. Sie können das Standardinstanzverzeichnis auch ändern, indem Sie den Wert der Umgebungsvariablen **DB2INSTPROF** ändern. Wenn Sie es zum Beispiel auf `c:\DB2PROFS` setzen wollen, gehen Sie wie folgt vor:
  - Setzen Sie **DB2INSTPROF** mithilfe des Befehls **db2set.exe -g** auf den Wert `c:\DB2PROFS`.
  - Führen Sie den Befehl **DB2ICRT.exe** aus, um die Instanz zu erstellen.
- Wenn Sie unter Windows-Betriebssystemen eine Instanz erstellen, sind die Standardpositionen für Datendateien des Benutzers - wie zum Beispiel Instanzverzeichnisse und die Datei `db2cli.ini` - die folgenden Verzeichnisse:
  - Unter Windows XP und Windows 2003: `Dokumente und Einstellungen\All Users\Application Data\IBM\DB2\`*`Name_der_Kopie`*
  - Unter den Betriebssystemen Windows 2008 und Windows Vista (und späteren Versionen): `Program Data\IBM\DB2\`*`Name_der_Kopie`*

  Dabei ist *Name_der_Kopie* der Name der DB2-Kopie.

  Anmerkung: Die Position der Datei `db2cli.ini` kann sich ändern, je nachdem, ob der Microsoft ODBC Driver Manager verwendet wird, welcher Typ von Datenquellennamen (DSN) verwendet wird, welcher Typ von Client oder Treiber installiert ist und ob die Registrierdatenbankvariable **DB2CLIINIPATH** definiert ist.

Aktualisieren von DB2-Kopien (Linux und UNIX)

Sie können eine vorhandene DB2-Kopie und alle Instanzen, die in dieser Kopie ausgeführt werden, aktualisieren. Außerdem haben Sie die Möglichkeit, die Installation einer neuen DB2-Kopie auszuwählen, und nach der Installation Instanzen für die Ausführung in dieser Kopie selektiv zu aktualisieren.

Vorbereitende Schritte

- Stellen Sie sicher, dass Sie über die Rootberechtigung verfügen.
- Laden Sie ein Fixpack herunter und dekomprimieren Sie es. Das Fixpack und die DB2-Kopie, die Sie aktualisieren wollen, müssen dasselbe Release-Level haben.

Informationen zu diesem Vorgang

Mit den Anweisungen in diesem Abschnitt können Sie Ihre DB2-Kopien von einer Fixpackstufe auf eine andere Fixpackstufe (mit demselben Versionsstand) aktualisieren oder zusätzliche Funktionalität installieren.

Wenn Sie Kopien von DB2 Version 8, Version 9.1, Version 9.5 oder Version 9.7 haben, können Sie diese Kopien nicht von früheren Releases auf DB2 Version 9.8 aktualisieren, sondern müssen ein Upgrade dieser Kopien durchführen.

Einschränkungen

- Das gleichzeitige Aktualisieren mehrerer DB2-Kopien ist nicht möglich. Zur Aktualisierung weiterer DB2-Kopien, die möglicherweise auf demselben Computer installiert sind, müssen Sie die Installation erneut ausführen.

Vorgehensweise

Gehen Sie zum Aktualisieren Ihrer DB2-Kopien wie folgt vor:

1. Melden Sie sich mit Rootberechtigung an.
2. Stoppen Sie alle DB2-Prozesse.
3. Aktualisieren Sie jede DB2-Kopie mit einer der folgenden Methoden:
   - Führen Sie den Befehl **installFixPack** aus, um eine vorhandene DB2-Kopie und alle Instanzen, die in dieser DB2-Kopie ausgeführt werden, zu aktualisieren. Mit diesem Befehl kann keine zusätzliche Funktionalität installiert werden.
   - Zur Installation einer neuen DB2-Kopie und zur selektiven Aktualisierung der Instanzen, die in einer vorhandenen DB2-Kopie ausgeführt werden, auf die neue Kopie nach der Installation führen Sie den Befehl **db2setup** aus und wählen **Neue Installation** in der Anzeige **Produkt installieren** aus. Zur Installation einer neuen Kopie können Sie auch eine Installation mit Antwortdatei unter Angabe einer neuen Position für den Installationspfad ausführen. Über alle diese Optionen können Sie auch zusätzliche Funktionalität installieren.
   - Wenn Sie einer vorhandenen DB2-Kopie Funktionalität hinzufügen wollen, wählen Sie **Mit vorhandener Installation arbeiten** in der Anzeige **Produkt installieren** aus. Wählen Sie anschließend die DB2-Kopie aus, die Sie mit der Aktion **Neue Funktion hinzufügen** aktualisieren wollen. Diese Aktion steht nur zur Verfügung, wenn die DB2-Kopie dasselbe Release-Level wie das Installationsimage hat. Zum Hinzufügen von Funktionalität können Sie auch eine Installation mit Antwortdatei oder den Befehl **db2_install** ausführen.

   Wichtig: Der Befehl **db2_install** ist veraltet und wird möglicherweise in einem zukünftigen Release entfernt. Verwenden Sie stattdessen den Befehl **db2setup** mit einer Antwortdatei.
4. Wenn Sie eine neue DB2-Kopie installiert haben, verwenden Sie den Befehl **db2iupdt**, um alle Instanzen zu aktualisieren, die in einer anderen DB2-Kopie desselben Release ausgeführt werden und unter der neuen Kopie ausgeführt werden sollen. Die folgende Tabelle zeigt einige Beispiele für das Aktualisieren von Instanzen:

Instanz	DB2-Kopie	Beispiele für das Aktualisieren zur Ausführung in anderer Kopie
db2inst1	/opt/IBM/db2/V9.1/	cd /opt/IBM/db2/V9.1_FP3/instance ./db2iupdt db2inst1

Instanz	DB2-Kopie	Beispiele für das Aktualisieren zur Ausführung in anderer Kopie
db2inst2	/opt/IBM/db2/V9.5FP2/	cd /home/db2/myV9.5_FP1/instance ./db2iupdt -D db2inst2[a]
db2inst3	/opt/IBM/db2/V9.7/	cd /home/db2/myV9.7/instance ./db2iupdt -k db2inst3[b]

Anmerkung:

a. Verwenden Sie den Parameter **-D**, um eine Instanz von einer Kopie eines späteren Release-Levels auf eine Kopie eines früheren Release-Levels zu aktualisieren.

b. Verwenden Sie den Parameter **-k**, um den aktuellen Instanztyp während der Aktualisierung auf einer DB2-Kopie zu halten, die ein höheres Level von Instanztyp hat. Wenn Sie von WSE auf ESE aktualisieren, wird der Instanztyp WSE bei einer Aktualisierung ohne diesen Parameter in ESE konvertiert.

Ergebnisse

Wenn Sie eine DB2-Kopie installiert oder aktualisiert haben, können Sie Instanzen, die in anderen DB2-Kopien desselben Release ausgeführt werden, immer zur Ausführung in dieser neuen DB2-Kopie aktualisieren, indem Sie den Befehl **db2iupdt** ausführen.

Aktualisieren von DB2-Kopien (Windows)

Sie können eine vorhandene DB2-Kopie und alle Instanzen, die in dieser Kopie ausgeführt werden, auf eine neue Fixpackstufe aktualisieren. Außerdem haben Sie die Möglichkeit, die Installation einer neuen DB2-Kopie auszuwählen, und nach der Installation Instanzen für die Ausführung in dieser Kopie selektiv zu aktualisieren.

Vorbereitende Schritte

- Stellen Sie sicher, dass Sie über die Berechtigung eines lokalen Administrators verfügen.
- Laden Sie ein Fixpack herunter und dekomprimieren Sie es. Das Fixpack und die DB2-Kopie, die Sie aktualisieren wollen, müssen dasselbe Release-Level haben.

Informationen zu diesem Vorgang

Mit den Anweisungen in diesem Abschnitt können Sie Ihre DB2-Kopien von einer Fixpackstufe auf eine andere Fixpackstufe (mit demselben Versionsstand) aktualisieren oder zusätzliche Funktionalität installieren.

Einschränkungen

- Sie können nur eine Instanz desselben Release von einer Kopie eines früheren Release-Levels auf eine Kopie eines späteren Release-Levels aktualisieren. Die Aktualisierung einer Instanz von einer Kopie eines späteren Release-Levels auf ein früheren Release-Level ist nicht möglich.
- Das gleichzeitige Aktualisieren mehrerer DB2-Kopien ist nicht möglich. Zur Aktualisierung weiterer DB2-Kopien, die möglicherweise auf demselben Computer installiert sind, müssen Sie die Installation erneut ausführen.

- Die Koexistenz eines DB2-Datenservers (32 Bit) und eines DB2-Datenservers (64 Bit) auf demselben Windows x64-Computer wird nicht unterstützt. Es ist nicht möglich, ein Upgrade direkt von einer 32-Bit-x64-Installation von DB2 der Version 8 auf eine 64-Bit-Installation der Version 9.8 durchzuführen.

Vorgehensweise

Gehen Sie zum Aktualisieren Ihrer DB2-Kopien wie folgt vor:

1. Melden Sie sich als Benutzer mit der Berechtigung eines lokalen Administrators an.
2. Stoppen Sie alle DB2-Instanzen, -Services und -Anwendungen.
3. Führen Sie die Datei `setup.exe` aus, um den DB2-Assistenten zur Installation einer DB2-Kopie zu starten. Sie haben die folgenden Auswahlmöglichkeiten:
   - Zur Aktualisierung einer vorhandenen DB2-Kopie und aller Instanzen, die in dieser DB2-Kopie ausgeführt werden, wählen Sie die Option **Mit vorhandener Installation arbeiten** in der Anzeige **Produkt installieren** aus. Wählen Sie anschließend die DB2-Kopie aus, die Sie mit der Aktion **Aktualisieren** aktualisieren wollen. Mit dieser Aktion können Sie keine zusätzliche Funktionalität installieren.
   - Zur Installation einer neuen DB2-Kopie und zur selektiven Aktualisierung der Instanzen, die in einer vorhandenen DB2-Kopie ausgeführt werden, auf die neue Kopie nach der Installation wählen Sie **Neue Installation** in der Anzeige **Produkt installieren** aus. Über diese Option können Sie auch zusätzliche Funktionalität installieren.
   - Wenn Sie einer vorhandenen DB2-Kopie Funktionalität hinzufügen wollen, wählen Sie **Mit vorhandener Installation arbeiten** in der Anzeige **Produkt installieren** aus. Wählen Sie anschließend die DB2-Kopie aus, die Sie mit der Aktion **Neue Funktion hinzufügen** aktualisieren wollen. Diese Aktion steht nur zur Verfügung, wenn die DB2-Kopie dasselbe Release-Level wie das Installationsimage hat.
4. Wenn Sie eine neue DB2-Kopie installiert haben, verwenden Sie den Befehl **db2iupdt**, um alle Instanzen zu aktualisieren, die in einer anderen DB2-Kopie desselben Release ausgeführt werden und unter der neuen Kopie ausgeführt werden sollen. Die folgende Tabelle zeigt einige Beispiele für das Aktualisieren von Instanzen:

Instanz	DB2-Kopie	Beispiele für das Aktualisieren zur Ausführung in anderer Kopie
db2inst1	`C:\Programme\IBM\SQLLIB_91\BIN`	`cd D:\Programme\IBM\SQLLIB_91_FP5\BIN` `db2iupdt db2inst1 /u: benutzername,kennwort`
db2inst2	`C:\Programme\IBM\SQLLIB_97\BIN`	`cd D:\Programme\IBM\SQLLIB_97\BIN` `db2iupdt db2inst2 /u: benutzername,kennwort`

Ergebnisse

Wenn Sie eine DB2-Kopie installiert oder aktualisiert haben, können Sie Instanzen, die in anderen DB2-Kopien desselben Release ausgeführt werden, immer zur Ausführung in dieser neuen DB2-Kopie aktualisieren, indem Sie den Befehl **db2iupdt** ausführen.

Gleichzeitiges Ausführen mehrerer Instanzen (Windows)

Sie können mehrere Instanzen gleichzeitig ausführen, entweder in derselben DB2-Kopie oder in unterschiedlichen DB2-Kopien.

Vorgehensweise

1. Gehen Sie unter Verwendung der Befehlszeile wie folgt vor, um mehrere Instanzen gleichzeitig in derselben DB2-Kopie auszuführen:
   a. Setzen Sie die Variable **DB2INSTANCE** auf den Namen der anderen Instanz, die Sie starten wollen, indem Sie folgenden Befehl eingeben:
      ```
      set db2instance=anderer_instanzname
      ```
   b. Geben Sie den Befehl **db2start** ein, um die Instanz zu starten.
2. Verwenden Sie eine der folgenden Methoden, wenn Sie mehrere Instanzen gleichzeitig in unterschiedlichen DB2-Kopien ausführen möchten:
   - Mithilfe des DB2-Befehlsfensters über **Start** > **Programme** > **IBM DB2** > *name_der_db2-kopie* > **Befehlszeilentools** > **DB2-Befehlsfenster**: Das Befehlsfenster ist bereits mit den richtigen Umgebungsvariablen für die jeweilige ausgewählte DB2-Kopie eingerichtet.
   - Mithilfe der Datei **db2envar.bat** über ein Befehlsfenster:
     a. Öffnen Sie ein Befehlsfenster.
     b. Führen Sie die Datei **db2envar.bat** aus, und verwenden Sie dabei den vollständig qualifizierten Pfad für die DB2-Kopie, die die Anwendung verwenden soll:
        ```
        installationsverz_der_db2-kopie\bin\db2envar.bat
        ```

   Verwenden Sie die Methode im oben stehenden Abschnitt ("Gleichzeitiges Ausführen mehrerer Instanzen in derselben DB2-Kopie"), wenn Sie zu einer bestimmten DB2-Kopie gewechselt haben, um die Instanzen zu starten.

Arbeiten mit Instanzen in derselben oder in anderen DB2-Kopien

Sie können mehrere Instanzen gleichzeitig in derselben DB2-Kopie oder in verschiedenen DB2-Kopien ausführen.

Informationen zu diesem Vorgang

Damit eine Instanz nicht auf die Datenbanken einer anderen Instanz zugreift, werden die Datenbankdateien für eine Instanz unter einem Verzeichnis erstellt, das den gleichen Namen wie die Instanz besitzt. Wenn zum Beispiel eine Datenbank auf dem Laufwerk `C:` für die Instanz `DB2` erstellt wird, werden die Datenbankdateien in einem Verzeichnis mit dem Namen `C:\DB2` erstellt. Analog werden bei der Erstellung einer Datenbank auf Laufwerk `C:` für die Instanz `TEST` die Datenbankdateien in einem Verzeichnis mit dem Namen `C:\TEST` erstellt. Standardmäßig entspricht der Wert dem Buchstaben des Laufwerks, in dem das DB2-Produkt installiert ist. Weitere Informationen finden Sie im Abschnitt zum Datenbankmanagerkonfigurationsparameter **dftdbpath**.

Vorgehensweise

- Wenn Sie mit Instanzen in derselben DB2-Kopie arbeiten möchten, müssen Sie wie folgt vorgehen:
  1. Führen Sie das Erstellen bzw. das Upgrade aller Instanzen in ein und derselben DB2-Kopie aus.
  2. Setzen Sie die Umgebungsvariable **DB2INSTANCE** auf den Namen der Instanz, mit der Sie arbeiten. Diese Aktion muss auftreten, bevor Sie Befehle für die Datenbank absetzen.
- Verwenden Sie eine der folgenden Methoden, wenn Sie mit einer Instanz in einem System mit mehreren DB2-Kopien arbeiten möchten:
  - Verwenden Sie das Befehlsfenster über **Start** > **Programme** > **IBM DB2** > ***name_der_db2-kopie*** > **Befehlszeilentools** > **Befehlsfenster**. Das Befehlsfenster ist bereits mit den richtigen Umgebungsvariablen für die jeweils ausgewählte DB2-Kopie eingerichtet.
  - Verwenden Sie `db2envar.bat` über ein Befehlsfenster:
    1. Öffnen Sie ein Befehlsfenster.
    2. Führen Sie die Datei `db2envar.bat` aus und verwenden Sie dabei den vollständig qualifizierten Pfad für die DB2-Kopie, die die Anwendung verwenden soll:

       installationsverz_der_db2-kopie`\bin\db2envar.bat`

Kapitel 3. Autonomic Computing - Übersicht

Die Autonomic Computing-Umgebung von DB2 bietet automatische Funktionen zur Selbstkonfiguration, zur Fehlerbehebung, Selbstoptimierung sowie zum Selbstschutz. Durch die Fähigkeit zur Erkennung eintretender Situationen und die Einleitung entsprechender Reaktionen verlagert Autonomic Computing die Verwaltungslast für eine IT-Umgebung von Datenbankadministratoren auf die Technologie.

Im Abschnitt „Automatische Funktionen" auf Seite 23 finden Sie eine allgemeine Übersicht über die Funktionen, aus denen sich die DB2-Autonomic-Computing-Umgebung zusammensetzt. Die folgende Tabelle enthält eine etwas detailliertere, nach Kategorien geordnete Übersicht über die Autonomic-Computing-Funktionen des Produkts:

Tabelle 2. Übersicht über die Informationen zu Autonomic Computing

Kategorie	Zugehörige Themen
Speicher mit automatischer Leistungsoptimierung	• „Speicherbelegung" in *Fehlerbehebung und Optimieren der Datenbankleistung* • „Speicher mit automatischer Leistungsoptimierung" in *Fehlerbehebung und Optimieren der Datenbankleistung* • „Speicher mit automatischer Leistungsoptimierung - Übersicht" in *Fehlerbehebung und Optimieren der Datenbankleistung* • „auto_maint - Automatische Verwaltung" auf Seite 844 • „db_storage_path - Pfad für dynamischen Speicher (Monitorelement)" in *Datenbanküberwachung - Handbuch und Referenz* • „num_db_storage_paths - Anzahl Pfade für dynamischen Speicher (Monitorelement)" in *Datenbanküberwachung - Handbuch und Referenz* • „tablespace_using_auto_storage - Verwendung des dynamischen Speichers (Monitorelement)" in *Datenbanküberwachung - Handbuch und Referenz* • „Konfigurieren von Speicher und der Zwischenspeicher" auf Seite 44 • „Konfiguration von Agenten, des Prozessmodells und des Speichers - Übersicht" auf Seite 47 • „Gemeinsam genutzte Tabelle für Dateikennungen" auf Seite 52 • „Ausführen von Bibliotheksfunktionen anderer Anbieter in Prozessen im abgeschirmten Modus" auf Seite 52 • „ADMIN_GET_MEM_USAGE (Tabellenfunktion) - Gesamtspeicherbelegung für Instanz abrufen" in *Administrative Routines and Views* • „Konfiguration des Agenten und des Prozessmodells" auf Seite 47 • „Konfigurieren von Datenbanken über mehrere Partitionen" auf Seite 50
Dynamischer Speicher	• „Datenbanken arbeiten standardmäßig mit dynamischem Speicher" auf Seite 53 • „Tabellenbereiche mit dynamischem Speicher" auf Seite 185 • „Automatische Änderung der Größe von DMS-Tabellenbereichen" auf Seite 181

Tabelle 2. Übersicht über die Informationen zu Autonomic Computing (Forts.)

Kategorie	Zugehörige Themen
Datenkomprimierung	• „Datenkomprimierung" auf Seite 53 – „Tabellenkomprimierung" auf Seite 339 – „Indexkomprimierung" auf Seite 496 – „Backupkomprimierung" in *Datenrecovery und hohe Verfügbarkeit - Handbuch und Referenz* • „Erstellen von Komprimierungswörterverzeichnissen auf Tabellenebene" auf Seite 351 • „Erstellung von Komprimierungswörterverzeichnissen bei LOAD-Operationen" in *Dienstprogramme für das Versetzen von Daten - Handbuch und Referenz*
Automatisches Datenbank-Backup	• „Automatisches Datenbank-Backup" in *Datenrecovery und hohe Verfügbarkeit - Handbuch und Referenz* • „Aktivieren des automatischen Backups" in *Datenrecovery und hohe Verfügbarkeit - Handbuch und Referenz* • „Entwickeln einer Backup- und Recoverystrategie" in *Datenrecovery und hohe Verfügbarkeit - Handbuch und Referenz*
Automatische Reorganisation	„Automatische Reorganisation" in *Fehlerbehebung und Optimieren der Datenbankleistung*
Automatische Statistikerfassung	• „Automatische Statistikerfassung" in *Fehlerbehebung und Optimieren der Datenbankleistung* • „Verwenden der automatischen Statistikerfassung" in *Fehlerbehebung und Optimieren der Datenbankleistung* • „Bei der automatischen Erstellung von Statistikdaten und der Profilermittlung verwendeter Speicher" in *Fehlerbehebung und Optimieren der Datenbankleistung* • „Aktivitätsprotokollierung für die automatische Statistikerfassung" in *Fehlerbehebung und Optimieren der Datenbankleistung*
Konfigurationsadvisor	• „Generieren von Konfigurationsempfehlungen" auf Seite 61 – „Optimieren von Konfigurationsparametern mit dem Konfigurationsadvisor" auf Seite 60 – „Beispiel: Anfordern von Konfigurationsempfehlungen mit dem Konfigurationsadvisor" auf Seite 61 – „AUTOCONFIGURE (Befehl)" in *Command Reference* – „Befehl AUTOCONFIGURE mit Verwendung der Prozedur ADMIN_CMD" in *Administrative Routines and Views* – „db2AutoConfig (API) - Auf Konfigurationsadvisor zugreifen" in *Administrative API Reference* • „Einstiegstipps für die Leistungsoptimierung" in *Fehlerbehebung und Optimieren der Datenbankleistung*

Tabelle 2. Übersicht über die Informationen zu Autonomic Computing (Forts.)

Kategorie	Zugehörige Themen
Diagnosemonitor	• „Diagnosemonitor" in *Datenbanküberwachung - Handbuch und Referenz* • „Prozesszyklus von Diagnoseanzeigern" in *Datenbanküberwachung - Handbuch und Referenz* – „Aktivieren der Diagnosealertbenachrichtigung" in *Datenbanküberwachung - Handbuch und Referenz* – „Konfigurieren von Diagnoseanzeigern über eine Clientanwendung" in *Datenbanküberwachung - Handbuch und Referenz* • „Diagnoseanzeiger - Zusammenfassung" in *Datenbanküberwachung - Handbuch und Referenz*
Drosselung von Dienstprogrammen	• „Drosselung von Dienstprogrammen" auf Seite 63 • „Asynchrone Indexbereinigung" in *Fehlerbehebung und Optimieren der Datenbankleistung* • „Asynchrone Indexbereinigung für MDC-Tabellen" in *Fehlerbehebung und Optimieren der Datenbankleistung* – „LIST UTILITIES (Befehl)" in *Command Reference* – „SET UTIL_IMPACT_PRIORITY (Befehl)" in *Command Reference* – „util_impact_lim - Richtlinie für Instanzauslastungswirkung" auf Seite 834 – „utility_priority - Dienstprogrammpriorität (Monitorelement)" in *Datenbanküberwachung - Handbuch und Referenz*

Automatische Funktionen

Automatische Funktionen unterstützen Sie bei der Verwaltung Ihres Datenbanksystems. Sie statten Ihr System mit der Möglichkeit aus, Selbstdiagnosen durchzuführen und Probleme vor dem Eintreten im Voraus zu erkennen. Zu diesem Zweck werden Echtzeitdaten analysiert und mit Langzeitproblemdaten verglichen. Einige automatische Tools können so konfiguriert werden, dass sie Änderungen an Ihrem System ohne Bedienereingriff durchführen und auf diese Weise Serviceunterbrechungen vermeiden helfen.

Beim Erstellen einer Datenbank werden einige der folgenden automatischen Funktionen standardmäßig aktiviert, während Sie andere manuell aktivieren müssen:

Speicher mit automatischer Leistungsoptimierung (nur Einzelpartitionsdatenbanken)
Die Funktion zur automatischen Speicherleistungsoptimierung vereinfacht die Aufgabe der Speicherkonfiguration. Diese Funktion reagiert auf signifikante Änderungen in der Auslastung, indem sie die Werte verschiedener Speicherkonfigurationsparameter sowie die Größen der Pufferpools automatisch und iterativ anpasst, um die Leistung zu optimieren. Die Speicheroptimierungsfunktion verteilt verfügbare Speicherressourcen dynamisch unter verschiedenen Speicherkonsumenten, zu denen die Sortierfunktion, der Paketcache, die Sperrenliste und Pufferpools gehören. Sie können die automatische Speicheroptimierung nach der Erstellung einer Datenbank inaktivieren, indem Sie den Datenbankkonfigurationsparameter **self_tuning_mem** auf den Wert OFF setzen.

Dynamischer Speicher
Die Funktion für dynamischen Speicher vereinfacht die Speicherverwaltung für Tabellenbereiche. Bei der Erstellung einer Datenbank geben Sie die

Speicherpfade für die Standardspeichergruppe an, in denen der Datenbankmanager Ihre Tabellenbereichsdaten speichert. Anschließend verwaltet der Datenbankmanager die Container und die Speicherzuordnung für die Tabellenbereiche, je nachdem, wie sie von Ihnen erstellt und mit Daten gefüllt werden. Dann können Sie auch neue Speichergruppen erstellen oder vorhandene Gruppen ändern.

Datenkomprimierung
Tabellen und Indizes können komprimiert werden, um Speicherplatz einzusparen. Die Komprimierung erfolgt vollkommen automatisch. Wenn Sie mithilfe der Klausel COMPRESS YES der Anweisungen CREATE TABLE, ALTER TABLE, CREATE INDEX oder ALTER INDEX angegeben haben, dass eine Tabelle bzw. ein Index komprimiert werden soll, sind keine weiteren Aktionen zur Verwaltung der Komprimierung mehr erforderlich. (Die Konvertierung einer vorhandenen nicht komprimierten Tabelle bzw. eines nicht komprimierten Index zur Verwendung der Komprimierung, macht eine Reorganisation (REORG) erforderlich, um vorhandene Daten zu komprimieren.) Temporäre Tabellen werden automatisch komprimiert. Indizes für komprimierte Tabellen werden standardmäßig ebenfalls automatisch komprimiert.

Automatische Datenbankbackups
Eine Datenbank kann durch eine Vielzahl möglicher Hard- und Softwarefehler unbrauchbar werden. Die Bereithaltung eines möglichst aktuellen Gesamtbackups Ihrer Datenbank ist ein integraler Bestandteil der Planung und Implementierung einer Strategie zur Wiederherstellung eines Systems nach einem Katastrophenfall. Verwenden Sie automatische Datenbankbackups im Rahmen Ihrer Wiederherstellungsstrategie, um den Datenbankmanager zu veranlassen, ordnungsgemäß und regelmäßig Backups Ihrer Datenbank durchzuführen.

Automatische Reorganisation
Nach zahlreichen Änderungen an Tabellendaten können die Tabelle und die zugehörigen Indizes fragmentiert werden. Logisch sequenzielle Daten können sich auf nicht sequenziellen Seiten befinden, sodass der Datenbankmanager für den Datenzugriff zusätzliche Leseoperationen ausführen muss. Der automatische Reorganisationsprozess bewertet in regelmäßigen Abständen Tabellen und Indizes, deren Statistiken aktualisiert wurden, um zu prüfen, ob eine Reorganisation erforderlich ist, und terminiert solche Operationen, wenn sie erforderlich sind.

Automatische Statistikerfassung
Die automatische Statistikerfassung unterstützt eine Verbesserung der Datenbankleistung, indem sie sicherstellt, dass aktuelle Tabellenstatistiken verfügbar sind. Der Datenbankmanager ermittelt, welche Statistiken für Ihre Auslastung erforderlich sind und welche Statistiken aktualisiert werden müssen. Statistiken können asynchron (im Hintergrund) oder synchron, durch Sammeln von Laufzeitstatistikdaten während der Kompilierung von SQL-Anweisungen, erfasst werden. Anschließend kann das DB2-Optimierungsprogramm einen Zugriffsplan auswählen, der auf präzisen Statistiken beruht. Sie können die automatische Statistikerfassung nach der Erstellung einer Datenbank inaktivieren, indem Sie den Datenbankkonfigurationsparameter **auto_runstats** auf den Wert OFF setzen. Die Echtzeitstatistikerfassung kann nur aktiviert werden, wenn die automatische Statistikerfassung aktiviert ist. Die Echtzeitstatistikerfassung wird durch den Konfigurationsparameter **auto_stmt_stats** gesteuert.

Konfigurationsadvisor

Wenn Sie eine Datenbank erstellen, wird dieses Tool automatisch ausgeführt, um die Datenbankkonfigurationsparameter und die Größe des Standardpufferpools (IBMDEFAULTBP) zu bestimmen und festzulegen. Die Werte werden auf der Basis der Systemressourcen und des für das System vorgesehenen Gebrauchs ausgewählt. Diese erste automatische Optimierung sorgt dafür, dass Ihre Datenbank eine bessere Leistung erreicht als eine äquivalente Datenbank, die Sie mit den Standardwerten erstellen könnten. Sie sorgt außerdem dafür, dass Sie im Anschluss an die Erstellung der Datenbank weniger Zeit für die Optimierung Ihres Systems aufwenden müssen. Sie können den Konfigurationsadvisor jederzeit (auch nachdem die Datenbanken mit Daten gefüllt wurden) ausführen, um das Tool empfohlene Werte für eine Gruppe von Konfigurationsparametern berechnen und optional anwenden zu lassen und die Leistung auf der Basis der aktuellen Kenndaten des Systems zu optimieren.

Diagnosemonitor

Der Diagnosemonitor ist ein serverseitiges Tool, das Situationen oder Änderungen in Ihrer Datenbankumgebung proaktiv überwacht, die zu Leistungseinbußen oder potenziellen Ausfällen führen können. Eine Reihe von Statusinformationen wird ohne jede Form aktiver Überwachung Ihrerseits erfasst. Wenn ein Risiko für den ordnungsgemäßen Betrieb festgestellt wird, werden Sie vom Datenbankmanager informiert und Sie erhalten Empfehlungen zur weiteren Verfahrensweise. Der Diagnosemonitor erfasst Informationen über das System mithilfe des Snapshot Monitors und verursacht keinerlei Leistungsbeeinträchtigung. Außerdem aktiviert er keine Snapshot Monitor-Schalter zur Erfassung von Informationen.

Drosselung von Dienstprogrammen

Diese Funktion reguliert die Leistungsbeeinträchtigung durch Wartungsdienstprogramme, sodass sie während der Produktionszeiten gleichzeitig ausgeführt werden können. Obwohl die *Richtlinie für die Auslastungswirkung* für gedrosselte Dienstprogramme standardmäßig definiert wird, müssen Sie die *Priorität der Auslastungswirkung* festlegen, wenn Sie ein Dienstprogramm gedrosselt ausführen wollen. Das Drosselungssystem stellt sicher, dass die gedrosselten Dienstprogramme so häufig wie möglich ausgeführt werden, ohne gegen die Richtlinie für die Auslastungswirkung zu verstoßen. Gegenwärtig können Operationen zur Statistikerfassung, Backup-Operationen, Datenumverteilungen und asynchrone Indexbereinigungen gedrosselt werden.

Automatische Verwaltung

Der Datenbankmanager stellt Funktionen zur automatischen Verwaltung zur Verfügung, um bei Bedarf Datenbankbackups, Statistikaktualisierungen und Reorganisationen von Tabellen und Indizes auszuführen. Die Ausführung von Verwaltungsaktivitäten an Ihren Datenbanken ist wichtig, um sicherzustellen, dass sie im Hinblick auf Leistung und Wiederherstellbarkeit optimal gepflegt werden.

Zur Verwaltung Ihrer Datenbank gehören einige oder alle der folgenden Aktivitäten:

- **Backups.** Wenn Sie ein Backup einer Datenbank durchführen, erstellt der Datenbankmanager eine Kopie der Daten in der Datenbank und speichert diese auf einem anderen Speichermedium für den Fall, dass die Originaldaten verloren gehen oder beschädigt werden. Automatische Datenbankbackups helfen bei der

Gewährleistung, dass Ihre Datenbank ordnungsgemäß und regelmäßig gesichert wird, ohne dass Sie sich um den Zeitpunkt des Backups kümmern oder die Syntax des Befehls **BACKUP** kennen müssen.

- **Datendefragmentierung (Tabellen- oder Indexreorganisation).** Diese Verwaltungsaktivität kann die Effizienz erhöhen, mit der der Datenbankmanager auf Ihre Tabellen zugreift. Die automatische Reorganisation verwaltet eine Offlinereorganisation von Tabellen und Indizes, ohne dass Sie sich darum zu kümmern brauchen, wann und wie Ihre Daten zu reorganisieren sind.
- **Datenzugriffsoptimierung (Statistikerfassung).** Der Datenbankmanager aktualisiert die Statistikdaten in den Systemkatalogen für die Daten in einer Tabelle, für die Daten in Indizes oder für beide Arten von Daten (in einer Tabelle und den zugehörigen Indizes). Mit diesen Statistiken bestimmt das Optimierungsprogramm, welcher Pfad für den Zugriff auf die Daten verwendet wird. Die automatische Statistikerfassung versucht, die Leistung der Datenbank durch die Pflege aktueller Tabellenstatistiken zu verbessern. Dies dient dazu, dem Optimierungsprogramm die Möglichkeit zu geben, einen Zugriffsplan auf der Grundlage exakter Statistikdaten auszuwählen.
- **Statistikprofilerstellung.** Die automatische Statistikprofilerstellung gibt Empfehlungen, wann und wie Tabellenstatistiken zu erfassen sind, indem sie veraltete, fehlende oder falsche Statistikdaten erkennt und Statistikprofile auf der Basis des Abfragefeedbacks generiert.

Die Feststellung, ob und wann Verwaltungsaktivitäten auszuführen sind, kann zeitaufwendig sein. Die automatischen Verwaltungsfunktionen nehmen Ihnen diese Aufgabe jedoch ab. Sie können die Aktivierung dieser automatischen Verwaltungsfunktionen einfach und flexibel mithilfe der Datenbankkonfigurationsparameter für die automatische Verwaltung steuern. Durch Festlegen der Datenbankkonfigurationsparameter für automatische Verwaltung können Sie Ihre Verwaltungsziele angeben. Der Datenbankmanager bestimmt anhand dieser Verwaltungsziele, ob die Verwaltungsaktivitäten ausgeführt werden müssen, und führt während des nächsten verfügbaren Verwaltungsfensters (ein von Ihnen definierter Zeitraum) nur die erforderlichen Verwaltungsaktivitäten aus.

Ab IBM Data Studio Version 3.1 kann der Taskassistent für Folgendes verwendet werden: Konfigurieren der automatischen Verwaltung. Taskassistenten führen durch den Prozess der Definition von Optionen, der Prüfung automatisch generierter Befehle für die jeweilige Task und der Ausführung dieser Befehle. Weitere Einzelheiten finden Sie in Verwalten von Datenbanken mit Taskassistenten.

Verwaltungsfenster

Ein Verwaltungsfenster ist ein von Ihnen definierter Zeitraum für die Ausführung automatischer Verwaltungsaktivitäten. Solche Verwaltungsaktivitäten sind Backups, Statistikerfassungen, Statistikprofilerstellungen und Reorganisationen. Ein Offlinefenster kann der Zeitraum sein, in dem der Zugriff auf eine Datenbank nicht verfügbar ist. Ein Onlinefenster kann der Zeitraum sein, in dem Benutzer die Verbindung zu einer Datenbank herstellen können.

Ein Verwaltungsfenster ist nicht dasselbe wie eine Taskzeitplanung. Während eines Verwaltungsfensters wird nicht unbedingt jede automatische Verwaltungsaktivität ausgeführt. Vielmehr bewertet der Datenbankmanager das System, um den Ausführungsbedarf für die einzelnen Verwaltungsaktivitäten zu bestimmen. Wenn die Verwaltungsanforderungen nicht erfüllt sind, wird die Verwaltungsaktivität ausgeführt. Befindet sich die Datenbank in einem guten Verwaltungszustand, wird die Verwaltungsaktivität nicht ausgeführt.

Sie müssen sich überlegen, wann die automatischen Verwaltungsaktivitäten ausgeführt werden sollen. Die automatischen Verwaltungsaktivitäten beanspruchen Ressourcen auf Ihrem System und können bei der Ausführung die Leistung Ihrer Datenbank beeinträchtigen. Einige dieser Aktivitäten schränken zudem den Zugriff auf Tabellen, Indizes und Datenbanken ein. Daher müssen Sie geeignete Zeitfenster angeben, während deren der Datenbankmanager Verwaltungsaktivitäten ausführen kann.

Offlineverwaltungsaktivitäten

Offlineverwaltungsaktivitäten (Offline-Datenbank-Backups sowie Tabellen- und Indexreorganisationen) sind Verwaltungsaktivitäten, die nur im Offlineverwaltungsfenster stattfinden können. Das Ausmaß, in dem der Benutzerzugriff eingeschränkt wird, ist von der jeweils ausgeführten Verwaltungsaktivität abhängig:

- Während eines Offline-Backups können Anwendungen keine Verbindung zur Datenbank herstellen. Alle gerade verbundenen Anwendungen werden zwangsweise getrennt.
- Während einer offline ausgeführten Tabellen- oder Indexreorganisation (Datendefragmentierung) können Anwendungen auf die Daten in den Tabellen zugreifen, diese jedoch nicht aktualisieren.

Offlineverwaltungsaktivitäten werden bis zum Ende ausgeführt, selbst wenn sie dabei über das angegebene Zeitfenster hinausgehen. Mit der Zeit erfasst der interne Zeitplanmechanismus, wie sich Jobausführungszeiten am besten abschätzen lassen. Wenn das Offlineverwaltungsfenster für eine bestimmte Datenbankbackup- oder Reorganisationsaktivität zu klein ist, startet die Planungsfunktion (Scheduler) den Job beim nächsten Mal nicht wieder und überlässt es dem Diagnosemonitor, eine Benachrichtigung über eine erforderliche Erweiterung des Offlineverwaltungsfensters auszugeben.

Onlineverwaltungsaktivitäten

Onlineverwaltungsaktivitäten (automatische Statistikerfassung und -profilerstellung, Onlineindexreorganisationen und Online-Datenbank-Backups) sind Verwaltungsaktivitäten, die nur im Onlineverwaltungsfenster stattfinden können. Während der Ausführung von Onlineverwaltungsaktivitäten können alle bereits verbundenen Anwendungen verbunden bleiben und auch neue Verbindungen hergestellt werden. Zur Minimierung der Auswirkungen auf das System werden Online-Datenbank-Backups sowie die automatische Statistikerfassung und Statistikprofilerstellung durch einen adaptiven Drosselmechanismus für Dienstprogramme gedrosselt.

Onlineverwaltungsaktivitäten werden bis zum Ende ausgeführt, selbst wenn sie dabei über das angegebene Zeitfenster hinausgehen.

Speicher mit automatischer Leistungsoptimierung

Seit DB2 Version 9 vereinfacht eine Speicheroptimierungsfunktion die Aufgabe der Speicherkonfiguration, indem sie automatisch Werte für verschiedene Speicherkonfigurationsparameter einstellt. Wenn sie aktiviert ist, verteilt die Speicheroptimierungsfunktion verfügbare Speicherressourcen dynamisch unter folgenden Speicherkonsumenten: Pufferpools, Sperrenspeicher, Paketcache und Sortierspeicher.

Die Optimierungsfunktion arbeitet innerhalb der Speicherbegrenzungen, die durch den Konfigurationsparameter **database_memory** definiert sind. Der Wert dieses Parameters kann ebenfalls automatisch optimiert werden. Wenn die automatische Speicheroptimierung aktiviert ist (d. h., wenn der Parameter **database_memory** den Wert AUTOMATIC hat), bestimmt die Optimierungsfunktion den Gesamtspeicher-

bedarf für die Datenbank und erhöht bzw. verringert die Menge an Speicher, die für gemeinsam genutzten Datenbankspeicher zugeordnet ist, abhängig von den aktuellen Anforderungen der Datenbank. Wenn zum Beispiel der aktuelle Bedarf der Datenbank hoch ist und ausreichend freier Speicher auf dem System zur Verfügung steht, wird mehr Speicher für den gemeinsam genutzten Datenbankspeicher zugeordnet. Wenn der Bedarf an Datenbankspeicher sinkt oder die Größe des freien Speichers auf dem System auf einen zu niedrigen Wert zurückgeht, wird ein Teil des gemeinsam genutzten Datenbankspeichers freigegeben.

Wenn der Konfigurationsparameter **database_memory** nicht auf AUTOMATIC gesetzt ist, verwendet die Datenbank die Größe an Speicher, die für diesen Parameter angegeben wurde, und verteilt sie nach Bedarf an die Speicherkonsumenten. Sie können die Speichergröße auf eine von zwei Arten angeben: durch Setzen des Parameters **database_memory** auf einen bestimmten numerischen Wert oder auf den Wert COMPUTED. Im letzteren Fall basiert der Gesamtspeicher auf der Summe der Anfangswerte der Datenbankzwischenspeicher beim Starten der Datenbank.

Sie können die automatische Speicheroptimierung für die Speicherkonsumenten auch wie folgt aktivieren:

- Für Pufferpools verwenden Sie die Anweisung ALTER BUFFERPOOL oder CREATE BUFFERPOOL (mit dem Schlüsselwort AUTOMATIC).
- Für den Sperrenspeicher verwenden Sie den Datenbankkonfigurationsparameter **locklist** oder **maxlocks** (mit dem Wert AUTOMATIC).
- Für den Paketcache verwenden Sie den Datenbankkonfigurationsparameter **pckcachesz** (mit dem Wert AUTOMATIC).
- Für den Sortierspeicher verwenden Sie den Datenbankkonfigurationsparameter **sheapthres_shr** oder **sortheap** (mit dem Wert AUTOMATIC).

Änderungen, die aus Operationen der automatischen Speicheroptimierung resultieren, werden in Protokolldateien für die Speicheroptimierung aufgezeichnet, die sich im Unterverzeichnis `stmmlog` befinden. Diese Protokolldateien enthalten Zusammenfassungen über den Ressourcenbedarf der einzelnen Speicherkonsumenten während bestimmter Optimierungsintervalle, die durch Zeitmarken in den Protokolleinträgen bestimmt werden.

Wenn nur wenig Speicher verfügbar ist, fallen die Leistungsvorteile durch die automatische Speicheroptimierung eher begrenzt aus. Da Optimierungsentscheidungen auf der Datenbankauslastung basieren, schränken Auslastungen mit rasch wechselnden Speicheranforderungen die Effektivität des Speichermanagers zur automatischen Leistungsoptimierung (STMM, Self-Tuning Memory Manager) ein. Wenn sich die Speichermerkmale Ihrer Auslastung ständig ändern, optimiert STMM weniger häufig und unter wechselnden Zielbedingungen. In diesem Szenario erreicht STMM keine absolute Konvergenz, sondern versucht stattdessen eine Hauptspeicherkonfiguration beizubehalten, die für die aktuelle Auslastung optimiert ist.

Speicher mit automatischer Leistungsoptimierung

Seit DB2 Version 9 vereinfacht eine Speicheroptimierungsfunktion die Aufgabe der Speicherkonfiguration, indem sie automatisch Werte für verschiedene Speicherkonfigurationsparameter einstellt. Wenn sie aktiviert ist, verteilt die Speicheroptimierungsfunktion verfügbare Speicherressourcen dynamisch unter folgenden Speicherkonsumenten: Pufferpools, Sperrenspeicher, Paketcache und Sortierspeicher.

Die Optimierungsfunktion arbeitet innerhalb der Speicherbegrenzungen, die durch den Konfigurationsparameter **database_memory** definiert sind. Der Wert dieses Parameters kann ebenfalls automatisch optimiert werden. Wenn die automatische Speicheroptimierung aktiviert ist (d. h., wenn der Parameter **database_memory** den Wert AUTOMATIC hat), bestimmt die Optimierungsfunktion den Gesamtspeicherbedarf für die Datenbank und erhöht bzw. verringert die Menge an Speicher, die für gemeinsam genutzten Datenbankspeicher zugeordnet ist, abhängig von den aktuellen Anforderungen der Datenbank. Wenn zum Beispiel der aktuelle Bedarf der Datenbank hoch ist und ausreichend freier Speicher auf dem System zur Verfügung steht, wird mehr Speicher für den gemeinsam genutzten Datenbankspeicher zugeordnet. Wenn der Bedarf an Datenbankspeicher sinkt oder die Größe des freien Speichers auf dem System auf einen zu niedrigen Wert zurückgeht, wird ein Teil des gemeinsam genutzten Datenbankspeichers freigegeben.

Wenn der Konfigurationsparameter **database_memory** nicht auf AUTOMATIC gesetzt ist, verwendet die Datenbank die Größe an Speicher, die für diesen Parameter angegeben wurde, und verteilt sie nach Bedarf an die Speicherkonsumenten. Sie können die Speichergröße auf eine von zwei Arten angeben: durch Setzen des Parameters **database_memory** auf einen bestimmten numerischen Wert oder auf den Wert COMPUTED. Im letzteren Fall basiert der Gesamtspeicher auf der Summe der Anfangswerte der Datenbankzwischenspeicher beim Starten der Datenbank.

Sie können die automatische Speicheroptimierung für die Speicherkonsumenten auch wie folgt aktivieren:

- Für Pufferpools verwenden Sie die Anweisung ALTER BUFFERPOOL oder CREATE BUFFERPOOL (mit dem Schlüsselwort AUTOMATIC).
- Für den Sperrenspeicher verwenden Sie den Datenbankkonfigurationsparameter **locklist** oder **maxlocks** (mit dem Wert AUTOMATIC).
- Für den Paketcache verwenden Sie den Datenbankkonfigurationsparameter **pckcachesz** (mit dem Wert AUTOMATIC).
- Für den Sortierspeicher verwenden Sie den Datenbankkonfigurationsparameter **sheapthres_shr** oder **sortheap** (mit dem Wert AUTOMATIC).

Änderungen, die aus Operationen der automatischen Speicheroptimierung resultieren, werden in Protokolldateien für die Speicheroptimierung aufgezeichnet, die sich im Unterverzeichnis `stmmlog` befinden. Diese Protokolldateien enthalten Zusammenfassungen über den Ressourcenbedarf der einzelnen Speicherkonsumenten während bestimmter Optimierungsintervalle, die durch Zeitmarken in den Protokolleinträgen bestimmt werden.

Wenn nur wenig Speicher verfügbar ist, fallen die Leistungsvorteile durch die automatische Speicheroptimierung eher begrenzt aus. Da Optimierungsentscheidungen auf der Datenbankauslastung basieren, schränken Auslastungen mit rasch wechselnden Speicheranforderungen die Effektivität des Speichermanagers zur automatischen Leistungsoptimierung (STMM, Self-Tuning Memory Manager) ein. Wenn sich die Speichermerkmale Ihrer Auslastung ständig ändern, optimiert STMM weniger häufig und unter wechselnden Zielbedingungen. In diesem Szenario erreicht STMM keine absolute Konvergenz, sondern versucht stattdessen eine Hauptspeicherkonfiguration beizubehalten, die für die aktuelle Auslastung optimiert ist.

Speicher mit automatischer Leistungsoptimierung - Übersicht

Der Speicher mit automatischer Leistungsoptimierung vereinfacht die Speicherkonfiguration, indem automatisch Werte für Speicherkonfigurationsparameter eingestellt und die Größe von Pufferpools gesteuert werden. Wenn sie aktiviert ist, verteilt die Speicheroptimierungsfunktion verfügbare Speicherressourcen dynamisch unter folgenden Speicherkonsumenten: Pufferpools, Sperrenspeicher, Paketcache und Sortierspeicher.

Der Speicher mit automatischer Leistungsoptimierung wird durch den Datenbankkonfigurationsparameter **self_tuning_mem** aktiviert.

Die folgenden für den Hauptspeicher relevanten Datenbankkonfigurationsparameter können automatisch optimiert werden:

- **database_memory** - Größe des gemeinsam genutzten Datenbankspeichers
- **locklist** - Maximaler Speicher für Sperrenliste
- **maxlocks** - Maximale Anzahl von Sperren vor Eskalation
- **pckcachesz** - Größe des Paketcache
- **sheapthres_shr** - Schwellenwert für Sortierspeicher für gemeinsame Sortiervorgänge
- **sortheap** - Sortierspeichergröße

Hauptspeicherzuordnung

Die Hauptspeicherzuordnung und die Hauptspeicherfreigabe finden zu verschiedenen Zeiten statt. Hauptspeicher kann einem bestimmten Speicherbereich zugeordnet werden, wenn ein bestimmtes Ereignis auftritt (z. B. die Herstellung einer Verbindung durch eine Anwendung) oder er kann infolge einer geänderten Konfigurationsparametereinstellung neu zugeordnet werden.

In Abb. 1 auf Seite 31 werden die unterschiedlichen Speicherbereiche gezeigt, die der Datenbankmanager für unterschiedliche Zwecke zuordnet, sowie die Konfigurationsparameter angegeben, mit denen Sie die Größe dieser Speicherbereiche steuern können. Beachten Sie, dass in einer Umgebung mit partitionierten Datenbanken jede Datenbankpartition über einen eigenen Bereich für den gemeinsam genutzten Speicher des Datenbankmanagers verfügt.

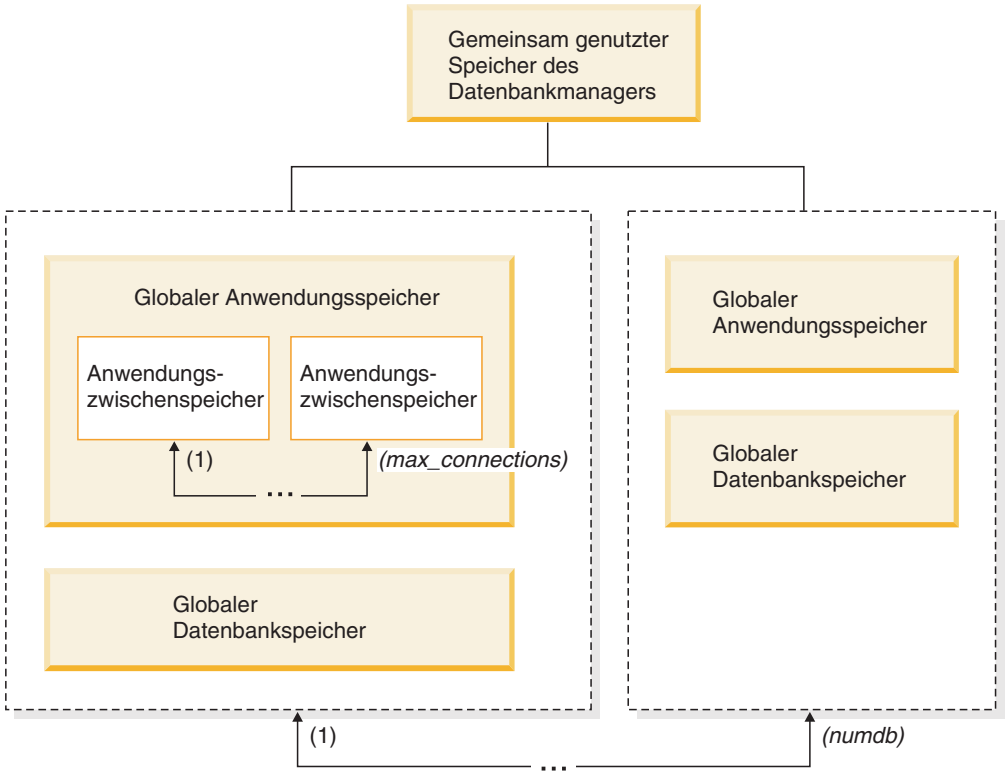


Abbildung 1. Vom Datenbankmanager zugeordnete Typen von Speicher

Speicher wird vom Datenbankmanager immer zugeordnet, wenn eines der folgenden Ereignisse auftritt:

Wenn der Datenbankmanager gestartet wird (`db2start`)

Der *gemeinsam genutzte Speicher des Datenbankmanagers* (auch als *gemeinsam genutzter Instanzspeicher* bezeichnet) bleibt so lange zugeordnet, bis der Datenbankmanager gestoppt wird (**`db2stop`**). Dieser Bereich enthält Informationen, die der Datenbankmanager zur Verwaltung von Aktivitäten für alle Datenbankverbindungen verwendet. Die Größe des gemeinsam genutzten Speichers des Datenbankmanagers wird von DB2 automatisch gesteuert.

Wenn eine Datenbank aktiviert oder zum ersten Mal eine Verbindung zu ihr hergestellt wird

Der *globale Datenbankspeicher* wird für alle Anwendungen verwendet, die eine Verbindung zur Datenbank herstellen. Die Größe des globalen Datenbankspeichers wird durch den Datenbankkonfigurationsparameter **`database_memory`** festgelegt. Standardmäßig hat dieser Parameter den AUTOMATIC, sodass DB2 die Anfangsgröße des Speichers, der der Datenbank zugeordnet wird, berechnen und die Größe des Datenbankspeichers während der Laufzeit automatisch je nach Bedarf der Datenbank konfigurieren kann.

Die folgenden Hauptspeicherbereiche können dynamisch angepasst werden:

- Pufferpools (unter Verwendung der Anweisung ALTER BUFFERPOOL)
- Datenbankzwischenspeicher (einschließlich Protokollpuffer)
- Zwischenspeicher für Dienstprogramme
- Paketcache
- Katalogcache

- Sperrenliste

Die Konfigurationsparameter **sortheap**, **sheapthres_shr** und **sheapthres** können auch dynamisch aktualisiert werden. Die einzige Einschränkung besteht darin, dass der Parameter **sheapthres** nicht dynamisch von 0 auf einen Wert größer 0 oder umgekehrt geändert werden kann.

Standardmäßig werden gemeinsame Sortieroperationen ausgeführt, und die Größe des gemeinsam genutzten Datenbankspeichers, der von Konsumenten des Sortierspeichers zu einem beliebigen Zeitpunkt verwendet werden kann, wird durch den Wert des Datenbankkonfigurationsparameters **sheapthres_shr** bestimmt. Private Sortieroperationen werden nur ausgeführt, wenn die partitionsinterne Parallelität, die Datenbankpartitionierung und der Verbindungskonzentrator inaktiviert sind und der Konfigurationsparameter **sheapthres** des Datenbankmanagers auf einen Wert ungleich null gesetzt ist.

Wenn eine Anwendung die Verbindung zu einer Datenbank herstellt
Jede Anwendung verfügt über einen eigenen *Anwendungszwischenspeicher*, der Teil des *globalen Anwendungsspeichers* ist. Sie können die Speicherkapazität, die jede einzelne Anwendung zuordnen kann, mithilfe des Datenbankkonfigurationsparameters **applheapsz** begrenzen. Sie können auch die Gesamtkapazität des Anwendungsspeichers mithilfe des Datenbankkonfigurationsparameters **appl_memory** begrenzen.

Wenn ein Agent erstellt wird
Der *private Agentenspeicher* wird für einen Agenten zugeordnet, wenn der Agent infolge einer Verbindungsanforderung oder einer neuen SQL-Anforderung in einer Umgebung mit partitionierten Datenbanken zugeordnet wird. Der private Agentenspeicher enthält Speicher, der nur von diesem speziellen Agenten verwendet wird. Wenn private Sortieroperationen aktiviert wurden, wird der private Sortierspeicher aus dem privaten Agentenspeicher zugeordnet.

Die folgenden Konfigurationsparameter begrenzen die Speicherkapazität, die für die einzelnen Typen von Speicherbereichen zugeordnet wird. Beachten Sie, dass dieser Speicher in einer Umgebung mit partitionierten Datenbanken in jeder Datenbankpartition zugeordnet wird.

numdb Dieser Konfigurationsparameter des Datenbankmanagers gibt die maximale Anzahl gleichzeitig aktiver Datenbanken an, die von verschiedenen Anwendungen verwendet werden können. Da jede Datenbank über einen eigenen globalen Speicherbereich verfügt, wächst die Menge an Speicher, die zugeordnet werden kann, wenn Sie den Wert dieses Parameters erhöhen.

maxappls
Dieser Datenbankkonfigurationsparameter definiert die maximale Anzahl von Anwendungen, die gleichzeitig eine Verbindung zu einer bestimmten Datenbank herstellen können. Der Wert dieses Parameters wirkt sich auf die Menge an Speicher aus, die sowohl für privaten Agentenspeicher als auch für den globalen Anwendungsspeicher der betreffenden Datenbank zugeordnet werden kann.

max_connections
Dieser Konfigurationsparameter des Datenbankmanagers begrenzt die Anzahl von Datenbankverbindungen (CONNECT) oder Instanzverbindungen (ATTACH), die auf den Datenserver zu einem gegebenen Zeitpunkt gleichzeitig zugreifen können.

max_coordagents
Dieser Konfigurationsparameter des Datenbankmanagers begrenzt die Anzahl von koordinierenden Agenten des Datenbankmanagers, die gleichzeitig für alle aktiven Datenbanken in einer Instanz (und pro Datenbankpartition in einer Umgebung mit partitionierten Datenbanken) vorhanden sein können. Zusammen mit den Parametern **maxappls** und **max_connections** begrenzt dieser Parameter die Größe des Speichers, der für den privaten Agentenspeicher und den globalen Anwendungsspeicher zugeordnet wird.

Sie können mit dem Speichertracker, der über den Befehl **db2mtrk** aufgerufen wird, die aktuelle Speicherzuordnung in der Instanz anzeigen. Darüber hinaus können Sie mithilfe der Tabellenfunktion ADMIN_GET_MEM_USAGE die Gesamtspeicherbelegung für die gesamte Instanz oder nur für eine einzelne Datenbankpartition ermitteln. Verwenden Sie die Tabellenfunktionen MON_GET_MEMORY_SET und MON_GET_MEMORY_POOL, um die aktuelle Speicherbelegung auf Instanz-, Datenbank- oder Anwendungsebene zu untersuchen.

Auf UNIX und Linux-Systemen können mit dem Befehl **ipcs** alle gemeinsam genutzten Speichersegmente aufgelistet werden; die Menge der genutzten Ressourcen wird jedoch nicht genau wiedergegeben. Alternativ zum Befehl **ipcs** kann der Befehl **db2mtrk** verwendet werden.

Interaktion und Einschränkungen von Speicherparametern

Obwohl Sie den Speicher mit automatischer Leistungsoptimierung aktivieren und die Standardeinstellung AUTOMATIC für die meisten auf Speicher bezogenen Konfigurationsparameter verwenden können, kann es sich als nützlich erweisen, die Einschränkungen der verschiedenen Speicherparameter und die Interaktionen zwischen ihnen zu kennen, um mehr Kontrolle über ihre Einstellungen zu haben und zu verstehen, warum Fehler aufgrund von Speicherknappheit unter bestimmten Bedingungen noch immer möglich sind.

Speichertypen

Der DB2-Datenbankmanager verwendet hauptsächlich zwei Typen von Speicher:

Leistungsspeicher
Dieser Speicher wird zur Verbesserung der Datenbankleistung verwendet. Der Leistungsspeicher wird vom Speichermanager für automatische Leistungsoptimierung (STMM - Self-Tuning Memory Manager) gesteuert und auf die verschiedenen Zwischenspeicher für die Leistung verteilt. Sie können den Konfigurationsparameter **database_memory** auf den Maximalwert an Leistungsspeicher setzen. Alternativ können Sie den Konfigurationsparameter **database_memory** auf den Wert AUTOMATIC setzen, um STMM die Verwaltung des gesamten Leistungsspeichers zu überlassen.

Funktionaler Speicher
Dieser Typ von Speicher wird von Anwendungsprogrammen verwendet. Sie können den Konfigurationsparameter **appl_memory** verwenden, um die maximale Menge an funktionalem Speicher bzw. Anwendungsspeicher zu steuern, die Serviceanwendungsanforderungen durch DB2-Datenbankagenten zugeordnet wird. Standardmäßig ist der Wert dieses Parameters auf AUTOMATIC gesetzt. Dies bedeutet, dass Anforderungen für funktionalen Speicher zulässig sind, solange Systemressourcen zur Verfügung stehen. Wenn Sie DB2-Datenbankprodukte mit Speichernutzungsbegrenzungen verwenden oder den Parameter **instance_memory** auf einen bestimmten Wert setzen, wird durch **instance_memory** eine Begrenzung festgelegt und

Anforderungen für funktionalen Speicher werden zugelassen, wenn sich die gesamte von der Datenbankpartition zugeordnete Speichermenge innerhalb des Grenzwerts des Parameters **instance_memory** bewegt.

Bevor die Einstellung AUTOMATIC verfügbar war, waren verschiedene Betriebssystem- und DB2-Tools verfügbar, mit denen die Speichermenge angezeigt werden konnte, die von unterschiedlichen Speichertypen, wie zum Beispiel vom gemeinsam genutzten Speicher, vom privaten Speicher, vom Pufferpoolspeicher, von Sperrenlisten, vom Sortierspeicher (von Zwischenspeichern) und so weiter, verwendet wurde. Es war jedoch nahezu unmöglich, den Gesamtspeicher anzuzeigen, der vom DB2-Datenbankmanager verwendet wurde. Wenn einer der Zwischenspeicher den Speichergrenzwert erreicht hatte, schlug eine Anweisung in einer Anwendung mit einer Fehlernachricht über Speicherknappheit fehl. Wenn Sie den Speicher für den betreffenden Zwischenspeicher vergrößerten und die Anwendung erneut ausführten, war es möglich, dass Sie einen Fehler aufgrund nicht ausreichenden Speichers für eine andere Anweisung in einem anderen Zwischenspeicher empfingen. Jetzt können Sie die festen oberen Grenzwerte für einzelne funktionale Zwischenspeicherbereiche aufheben, indem Sie die Standardeinstellung AUTOMATIC für Konfigurationsparameter verwenden.

Falls erforderlich (um Szenarios zu vermeiden, in denen eine Datenbankanwendung mit schlechter Leistung extrem große Speichermengen benötigt), können Sie einen Grenzwert auf den Gesamtanwendungsspeicher auf der Datenbankebene anwenden, indem Sie den Konfigurationsparameter **appl_memory** verwenden. Sie können einen Grenzwert für einen einzelnen Zwischenspeicher auch anwenden, indem Sie den Datenbankkonfigurationsparameter für diesen Zwischenspeicher von der Einstellung AUTOMATIC in einen festen Wert ändern. Wenn alle Konfigurationsparameter für alle funktionalen Zwischenspeicher auf AUTOMATIC gesetzt sind und mit dem Parameter **instance_memory** eine Begrenzung festgelegt ist, ist die Einstellung des Konfigurationsparameters **instance_memory** die einzige Begrenzung für die Anwendungsspeichernutzung. Wenn Sie auch den Konfigurationsparameter **instance_memory** auf AUTOMATIC setzen und ein DB2-Datenbankprodukt mit Speichernutzungsbegrenzung verwenden, bestimmt der DB2-Datenbankmanager die obere Grenze der Speichernutzung automatisch.

Sie können die Gesamtgröße des belegten Instanzspeichers und die aktuelle Nutzung des Parameters **instance_memory** ohne großen Aufwand mithilfe des Befehls **db2pd -dbptnmem** oder der Tabellenfunktion ADMIN_GET_MEM_USAGE anzeigen.

Interaktionen zwischen Hauptspeicherkonfigurationsparametern

Wenn der Speichermanager für automatische Leistungsoptimierung (STMM) aktiv ist und die automatische Leistungsoptimierung des Datenbankspeichers aktiviert ist (**database_memory** ist auf AUTOMATIC gesetzt), prüft STMM den freien Speicher, der auf dem System verfügbar ist, und stellt automatisch fest, wie viel Speicher für leistungsorientierte Zwischenspeicher dediziert werden sollte, um eine optimale Leistung zu erzielen. Alle Leistungszwischenspeicher tragen zur Gesamtgröße von **database_memory** bei. Neben dem Leistungsspeicherbedarf wird etwas Speicher benötigt, um den Betrieb und die Integrität des DB2-Datenbankmanagers sicherzustellen. Die Differenz zwischen dem Speicherbereich, der vom Parameter **instance_memory** verwendet wird, und dem Speicherbereich, der für diese beiden Speicherkonsumenten erforderlich ist, ist für den Anwendungsspeicher (**appl_memory**) verfügbar. Der funktionale Speicher für Anwendungsprogramme wird dann nach Bedarf zugeordnet. Wenn keine Begrenzung durch den Parameter **instance_memory** festgelegt wird, bestehen keine weiteren Einschränkungen dafür, wie viel Speicher eine einzelne Anwendung zuordnen kann.

Abhängig von der Konfiguration fragt STMM regelmäßig ab, wie viel freier Systemspeicher und wie viel freier Speicher für den Instanzspeicher (**instance_memory**) verbleibt, wenn eine Begrenzung durch den Parameter **instance_memory** besteht. Zur Vermeidung von Anwendungsfehlern ordnet STMM Anwendungsanforderungen höhere Priorität als Leistungskriterien ein. Falls erforderlich, senkt STMM die Leistung durch Verringern des Speicherbereichs, der für Leistungszwischenspeicher verfügbar ist, sodass genügend freier Systemspeicher und Instanzspeicher (**instance_memory**) zur Erfüllung von Anwendungsspeicheranforderungen zur Verfügung gestellt wird. Nach Beendigung von Anwendungen wird der belegte Speicher freigegeben und kann sofort durch andere Anwendungen wiederverwendet oder zur Verwendung als Datenbankspeicher (**database_memory**) von STMM freigegeben werden. Sinkt die Leistung des Datenbanksystems in Zeiträumen starker Anwendungsaktivität unter ein akzeptables Maß, kann es nützlich sein, entweder durch Steuermechanismen festzulegen, wie viele Anwendungen im Datenbankmanager zulässig sind (mithilfe des Verbindungskonzentrators oder der neuen Workload-Manager-Komponente von DB2 Version 9.5), oder dem System zusätzliche Speicherressourcen hinzuzufügen.

Aktivieren des Speichers mit automatischer Leistungsoptimierung

Der Speicher mit automatischer Leistungsoptimierung vereinfacht die Speicherkonfiguration, indem automatisch Werte für Speicherkonfigurationsparameter eingestellt und die Größe von Pufferpools gesteuert werden.

Informationen zu diesem Vorgang

Wenn sie aktiviert ist, verteilt die Speicheroptimierungsfunktion verfügbare Speicherressourcen dynamisch auf mehrere Speicherkonsumenten, zu denen Pufferpools, der Sperrenspeicher, der Paketcache und der Sortierspeicher gehören.

Vorgehensweise

1. Aktivieren Sie die automatische Speicheroptimierung für die Datenbank, indem Sie den Datenbankkonfigurationsparameter **self_tuning_mem** mithilfe des Befehls **UPDATE DATABASE CONFIGURATION** oder der API db2CfgSet auf den Wert `ON` setzen.
2. Zur Aktivierung der automatischen Optimierungsfunktion für Speicherbereiche, die durch Speicherkonfigurationsparameter gesteuert werden, setzen Sie die relevanten Konfigurationsparameter mithilfe des Befehls **UPDATE DATABASE CONFIGURATION** oder der API db2CfgSet auf den Wert `AUTOMATIC`.
3. Zur Aktivierung der automatischen Optimierungsfunktion für einen Pufferpool setzen Sie die Pufferpoolgröße mithilfe der Anweisung CREATE BUFFERPOOL oder ALTER BUFFERPOOL auf den Wert `AUTOMATIC`. In einer Umgebung mit partitionierten Datenbanken sollte dieser Pufferpool keine Einträge in der Katalogsicht SYSCAT.BUFFERPOOLDBPARTITIONS haben.

Ergebnisse

Anmerkung:

1. Da der Speicher mit automatischer Leistungsoptimierung zwischen verschiedenen Speicherkonsumenten verteilt wird, muss für mindestens zwei Hauptspeicherbereiche gleichzeitig die automatische Optimierung zu einem Zeitpunkt aktiviert sein, zum Beispiel für den Sperrenspeicher und den gemeinsam genutzten Datenbankspeicher. Die Speicheroptimierungsfunktion optimiert den

Hauptspeicher im System aktiv (der Datenbankkonfigurationsparameter **self_tuning_mem** hat den Wert ON), wenn eine der folgenden Bedingungen zutrifft:

- Ein Konfigurationsparameter oder eine Pufferpoolgröße ist auf AUTOMATIC gesetzt und der Datenbankkonfigurationsparameter **database_memory** ist entweder auf einen numerischen Wert oder auf AUTOMATIC gesetzt.
- Beliebige zwei der Parameter **locklist**, **sheapthres_shr**, **pckcachesz** oder der Pufferpoolgröße sind auf AUTOMATIC gesetzt.
- Der Datenbankkonfigurationsparameter **sortheap** ist auf AUTOMATIC gesetzt.

2. Der Wert des Datenbankkonfigurationsparameters **locklist** wird zusammen mit dem Datenbankkonfigurationsparameter **maxlocks** optimiert. Die Inaktivierung der automatischen Optimierung des Parameters **locklist** inaktiviert automatisch auch die automatische Optimierung des Parameters **maxlocks** und die Aktivierung der automatischen Optimierung des Parameters **locklist** aktiviert automatisch auch die automatische Optimierung des Parameters **maxlocks**.
3. Die automatische Optimierung des Datenbankkonfigurationsparameters **sortheap** oder **sheapthres_shr** ist nur zulässig, wenn der Konfigurationsparameter **sheapthres** des Datenbankmanagers auf den Wert 0 gesetzt ist.
4. Der Wert des Parameters **sortheap** wird zusammen mit dem Parameter **sheapthres_shr** optimiert. Die Inaktivierung der automatischen Optimierung des Parameters **sortheap** inaktiviert automatisch auch die automatische Optimierung des Parameters **sheapthres_shr** und die Aktivierung der automatischen Optimierung des Parameters **sheapthres_shr** aktiviert automatisch auch die automatische Optimierung des Parameters **sortheap**.
5. Die automatische Speicheroptimierungsfunktion wird nur auf dem primären HADR-Server (High Availability Disaster Recovery) ausgeführt. Wenn die automatische Speicheroptimierung auf einem HADR-System aktiviert wird, wird sie nie auf dem sekundären Server ausgeführt, und sie wird auch nur dann auf dem primären Server ausgeführt, wenn die Konfiguration ordnungsgemäß eingestellt ist. Wenn die HADR-Datenbankrollen vertauscht werden, wird die Funktion der automatischen Speicheroptimierung ebenfalls übertragen, sodass sie auf dem neuen primären Server ausgeführt wird. Nach dem Starten der Primärdatenbank oder dem Wechsel eines Systems von einer Bereitschaftsdatenbank zu einer Primärdatenbank durch Funktionsübernahme wird die EDU (Engine Dispatchable Unit, von der Steuerkomponente zuteilbare Einheit) der automatischen Speicheroptimierungsfunktion (STMM, Self-Tuning Memory Manager) möglicherweise erst gestartet, wenn der erste Client eine Verbindung herstellt.

Inaktivieren des Speichers mit automatischer Leistungsoptimierung

Die automatische Speicheroptimierung kann für die gesamte Datenbank oder für einen oder mehrere Konfigurationsparameter bzw. Pufferpools inaktiviert werden.

Informationen zu diesem Vorgang

Wenn die automatische Speicheroptimierung für die gesamte Datenbank inaktiviert wird, bleiben die Speicherkonfigurationsparameter und Pufferpools, die auf AUTOMATIC gesetzt sind, für die automatische Optimierung aktiviert, jedoch behalten die Speicherbereiche ihre aktuelle Größe.

Vorgehensweise

1. Inaktivieren Sie die automatische Speicheroptimierung für die Datenbank, indem Sie den Datenbankkonfigurationsparameter **self_tuning_mem** mithilfe des Befehls **UPDATE DATABASE CONFIGURATION** oder der API db2CfgSet auf den Wert OFF setzen.
2. Zur Inaktivierung der automatischen Optimierungsfunktion für Speicherbereiche, die durch Speicherkonfigurationsparameter gesteuert werden, setzen Sie die relevanten Konfigurationsparameter mithilfe des Befehls **UPDATE DATABASE CONFIGURATION** oder der API db2CfgSet auf den Wert MANUAL.
3. Zur Inaktivierung der automatischen Optimierungsfunktion für einen Pufferpool setzen Sie die Pufferpoolgröße mithilfe der Anweisung ALTER BUFFERPOOL auf einen bestimmten Wert.

Ergebnisse

Anmerkung:

- In einigen Fällen kann ein Speicherkonfigurationsparameter für die automatische Optimierungsfunktion nur dann aktiviert werden, wenn ein anderer Speicherkonfigurationsparameter ebenfalls aktiviert ist. Dies bedeutet zum Beispiel, dass eine Inaktivierung der automatischen Speicheroptimierung für die Datenbankkonfigurationsparameter **locklist** oder **sortheap** die automatische Speicheroptimierung für die Datenbankparameter **maxlocks** bzw. **sheapthres_shr** inaktiviert.

Ermitteln der Speicherkonsumenten mit aktivierter automatischer Leistungsoptimierung

Sie können die Einstellungen für den Speicher mit automatischer Leistungsoptimierung anzeigen, die von Konfigurationsparametern gesteuert werden oder für Pufferpools gelten.

Informationen zu diesem Vorgang

Es ist wichtig zu beachten, dass die Reaktionsfähigkeit der Speicheroptimierungsfunktion durch die Zeit eingeschränkt wird, die zur Änderung der Größe eines Speicherkonsumenten erforderlich ist. Zum Beispiel kann die Verringerung der Größe eines Pufferpools ein längerer Prozess sein und die Leistungsvorteile durch eine Verkleinerung des Pufferpoolspeichers zugunsten einer Erweiterung des Sortierspeichers können vielleicht nicht sofort realisiert werden.

Vorgehensweise

- Zum Anzeigen der Einstellungen für Konfigurationsparameter können Sie eine der folgenden Methoden verwenden:
  - Verwenden Sie den Befehl **GET DATABASE CONFIGURATION** und geben Sie den Parameter **SHOW DETAIL** an.

    Die Speicherkonsumenten, für die die automatische Optimierung aktiviert werden kann, werden in der Ausgabe wie folgt zusammengruppiert:

```
Beschreibung                                    Parameter    Aktueller Wert     Verzögerter Wert
-------------------------------------------------------------------------------------------------
Speicher mit automatischer Leistungsoptimierung          (SELF_TUNING_MEM) = ON (Aktiv)         ON
Größe des gemeinsamen Datenbankspeichers (4 KB)          (DATABASE_MEMORY) = AUTOMATIC(37200)   AUTOMATIC(37200)
Max. Speicher für Sperrenliste (4 KB)                           (LOCKLIST) = AUTOMATIC(7456)    AUTOMATIC(7456)
Anzahl der Sperrenlisten pro Anwend. (in %)                     (MAXLOCKS) = AUTOMATIC(98)      AUTOMATIC(98)
 Größe des Paketcache (4 KB)                                 (PCKCACHESZ) = AUTOMATIC(5600)    AUTOMATIC(5600)
Sortierspeicherschwelle für gemeinsame Sortierungen (4 KB)(SHEAPTHRES_SHR) = AUTOMATIC(5000)    AUTOMATIC(5000)
Zwischenspeicher für Sortierlisten (4 KB)                       (SORTHEAP) = AUTOMATIC(256)     AUTOMATIC(256)
```

– Verwenden Sie die Anwendungsprogrammierschnittstelle db2CfgGet.

  Die folgenden Werte werden zurückgegeben:

  ```
  SQLF_OFF                  0
  SQLF_ON_ACTIVE            2
  SQLF_ON_INACTIVE          3
  ```

  SQLF_ON_ACTIVE gibt an, dass die automatische Optimierung aktiviert und aktiv ist, während SQLF_ON_INACTIVE anzeigt, dass die automatische Optimierung zwar aktiviert, jedoch zurzeit nicht aktiv ist.

- Zum Anzeigen der Einstellungen für die automatische Optimierungsfunktion für Pufferpools können Sie eine der folgenden Methoden verwenden:
  – Zum Abrufen einer Liste der Pufferpools, für die die automatische Optimierung aktiviert ist, verwenden Sie die folgende Abfrage:

    ```
    SELECT BPNAME, NPAGES FROM SYSCAT.BUFFERPOOLS
    ```

    Wenn die automatische Optimierung für einen Pufferpool aktiviert ist, hat das Feld NPAGES in der Sicht SYSCAT.BUFFERPOOLS für den betreffenden Pufferpool den Wert -2. Wenn die automatische Optimierung inaktiviert ist, enthält das Feld NPAGES die aktuelle Größe des Pufferpools.

  – Zur Ermittlung der aktuellen Größe von Pufferpools, für die die automatische Optimierung aktiviert wurde, verwenden Sie den Befehl **GET SNAPSHOT** und untersuchen die aktuelle Größe der Pufferpools (den Wert des Monitorelements **bp_cur_buffsz**):

    ```
    GET SNAPSHOT FOR BUFFERPOOLS ON datenbankaliasname
    ```

    Eine Anweisung ALTER BUFFERPOOL, die die Größe eines Pufferpools in einer bestimmten Datenbankpartition angibt, erstellt einen Ausnahmeeintrag (bzw. aktualisiert einen vorhandenen Eintrag) für diesen Pufferpool in der Katalogsicht SYSCAT.BUFFERPOOLDBPARTITIONS. Wenn ein Ausnahmeeintrag für einen Pufferpool vorhanden ist, wird dieser Pufferpool an Operationen zur automatischen Optimierung nicht beteiligt, wenn die Standardpufferpoolgröße auf den Wert AUTOMATIC gesetzt ist.

Speicher mit automatischer Leistungsoptimierung in Umgebungen mit partitionierten Datenbanken

Wenn die automatische Speicheroptimierungsfunktion in Umgebungen mit partitionierten Datenbanken verwendet wird, bestimmen einige wenige Faktoren, ob die Funktion das System geeignet optimiert.

Wenn der Speicher mit automatischer Leistungsoptimierung für partitionierte Datenbanken aktiviert wird, wird eine Datenbankpartition als Optimierungspartition bestimmt. Alle Entscheidungen bezüglich der Speicheroptimierung werden auf der Basis der Speicher- und Auslastungsmerkmale dieser Datenbankpartition getroffen. Wenn Optimierungsentscheidungen in dieser Partition getroffen werden, werden die Speicheranpassungen an die anderen Datenbankpartitionen verteilt, um sicherzustellen, dass alle Datenbankpartitionen ähnliche Konfigurationen behalten.

Das auf einer Optimierungspartition basierende Modell geht davon aus, dass die Funktion nur verwendet wird, wenn alle Datenbankpartitionen ähnliche Speicheranforderungen haben. Beachten Sie die folgenden Richtlinien bei der Entscheidung, ob die automatische Speicheroptimierung für eine partitionierte Datenbank aktiviert werden sollte.

Fälle, in denen die automatische Speicheroptimierung für partitionierte Datenbanken empfohlen wird

Wenn alle Datenbankpartitionen ähnliche Speicheranforderungen haben und auf ähnlicher Hardware betrieben werden, kann die automatische Speicheroptimierung ohne Modifikationen aktiviert werden. Solche Typen von Umgebungen haben die folgenden gemeinsamen Merkmale:

- Alle Datenbankpartitionen befinden sich auf identischer Hardware und mehrere logische Datenbankpartitionen sind gleichmäßig auf mehrere physische Datenbankpartitionen verteilt.
- Es ist eine perfekte oder nahezu perfekte Verteilung von Daten vorhanden.
- Auslastungen werden gleichmäßig über Datenbankpartitionen verteilt. Das heißt, keine Datenbankpartition hat höheren Speicherbedarf für einen oder mehrere Zwischenspeicherbereiche als irgendeine der anderen Datenbankpartitionen.

Wenn in einer solchen Umgebung alle Datenbankpartitionen gleich konfiguriert sind, sorgt die automatische Speicheroptimierung für eine ordnungsgemäße Konfiguration des Systems.

Fälle, in denen die automatische Speicheroptimierung für partitionierte Datenbanken unter Vorkehrungen empfohlen wird

In Fällen, in denen die meisten Datenbankpartitionen in einer Umgebung ähnliche Speicheranforderungen haben und auf ähnlicher Hardware betrieben werden, kann die automatische Speicheroptimierung eingesetzt werden, solange die Anfangskonfiguration mit Sorgfalt erfolgt. Solche Systeme haben möglicherweise nur eine Gruppe von Datenbankpartitionen für Daten und eine wesentlich kleinere Gruppe von Koordinatorpartitionen und Katalogpartitionen. In solchen Umgebungen kann es von Vorteil sein, die Koordinatorpartitionen und Katalogpartitionen anders zu konfigurieren als die Datenbankpartitionen mit den Daten.

Die automatische Speicheroptimierung sollte in allen Datenbankpartitionen aktiviert werden, die Daten enthalten, wobei eine dieser Datenbankpartitionen als Optimierungspartition vorgesehen werden sollte. Da die Koordinatorpartitionen und die Katalogpartitionen verschieden konfiguriert sein können, sollte außerdem die automatische Speicheroptimierung in diesen Partitionen inaktiviert werden. Zur Inaktivierung der automatischen Speicheroptimierung in den Koordinator- und Katalogpartitionen setzen Sie den Datenbankkonfigurationsparameter **self_tuning_mem** in diesen Partitionen auf den Wert OFF.

Fälle, in denen die automatische Speicheroptimierung für partitionierte Datenbanken nicht empfohlen wird

Wenn der Speicherbedarf für die einzelnen Datenbankpartitionen unterschiedlich ist oder verschiedene Datenbankpartitionen auf erheblich unterschiedlicher Hardware betrieben werden, ist es eine empfohlene Methode, die Funktion der automatischen Speicheroptimierung zu inaktivieren. Sie können die Funktion inaktivieren, indem Sie den Datenbankkonfigurationsparameter **self_tuning_mem** in allen Partitionen auf den Wert OFF setzen.

Vergleich des Speicherbedarfs verschiedener Datenbankpartitionen

Die beste Methode zur Bestimmung, ob die Speicheranforderungen verschiedener Datenbankpartitionen ausreichend ähnlich sind, ist die Verwendung des Snapshot Monitors. Wenn die folgenden Monitorelemente in allen Datenbankpartitionen ähn-

liche Werte liefern (mit Abweichungen unter 20 %), können die Datenbankpartitionen als ausreichend ähnlich betrachtet werden.

Erfassen Sie die folgenden Daten, indem Sie den Befehl `get snapshot for database on <datenbankname>` ausführen:

```
Aktuelle Sperren                                                        = 0
Warten bei Sperren                                                      = 0
Wartezeit der Datenbank bei Sperren (ms)                                = 0
Verwendeter Speicher für Sperrenlisten (Byte)                           = 4968
Sperreneskalationen                                                     = 0
Exklusive Sperreneskalationen                                           = 0

Gesamter zugeordneter gemeinsamer Sortierspeicher                       = 0
Obere Grenze für gemeinsamen Sortierspeicher                            = 0
Sortiervorgang nach Schwellenwertüberschreitung (gemeinsam genutzter Speicher) = 0
Überläufe bei Sortierung                                                = 0

Suchoperationen im Paket-Cache                                          = 13
Einfügungen im Paket-Cache                                              = 1
Überläufe des Paket-Caches                                              = 0
Obere Grenze für Paket-Cache (Byte)                                     = 655360

Anzahl Hash-Joins                                                       = 0
Anzahl Hash-Schleifen                                                   = 0
Anzahl Überläufe von Hash-Joins                                         = 0
Anzahl kleiner Überläufe von Hash-Joins                                 = 0
Hash-Joins nach Schwellenwertüberschreitung (gemeinsam genutzter Speicher)     = 0

Anzahl OLAP-Funktionen                                                  = 0
Anzahl Überläufe von OLAP-Funktionen                                    = 0
Aktive OLAP-Funktionen                                                  = 0
```

Erfassen Sie die folgenden Daten, indem Sie den Befehl `get snapshot for bufferpools on <datenbankname>` ausführen:

```
Logische Lesevorgänge im Pufferpool                                     = 0
Physische Lesevorgänge im Pufferpool                                    = 0
Logische Lesevorgänge im Pufferpoolindex                                = 0
Physische Lesevorgänge im Pufferpoolindex                               = 0
Gesamtzeit der Lesevorgänge im Pufferpool  (ms)                         = 0
Gesamtzeit der Schreibvorgänge im Pufferpool (ms)                       = 0
```

Speicher mit automatischer Leistungsoptimierung in einer DB2 pureScale-Umgebung

Wenn die Funktion für Speicher mit automatischer Leistungsoptimierung in einer DB2 pureScale-Umgebung aktiviert ist, überwacht das Optimierungs-Member die Speicherkonfiguration und gibt alle Konfigurationsänderungen an alle anderen Member weiter.

Wenn die Funktion für Speicher mit automatischer Leistungsoptimierung in einer DB2 pureScale-Umgebung aktiviert ist, überwacht ein einzelnes Member (das Optimierungs-Member) die Speicherkonfiguration und gibt alle Konfigurationsänderungen an alle anderen Member weiter, um eine konsistente Konfiguration über alle Member in der Instanz hinweg sicherzustellen.

Wenn in einer DB2 pureScale-Umgebung das Optimierungs-Member als -1 angegeben ist, wird dieses Member bei jeder Aktivierung der Datenbank per Zufallsprinzip ausgewählt. Wenn außerdem das Member, auf dem die Optimierung ausgeführt wird, inaktiviert wird, startet die Optimierung automatisch auf einem anderen Member, auf dem dieser Vorgang aktuell möglich ist. Damit eine Optimie-

rung auf einem Member ausgeführt werden kann, muss die Datenbank auf diesem Member aktiv und der Parameter **self_tuning_mem** für dieses Member auf den Wert ON gesetzt sein.

- Um zu ermitteln, welches Member aktuell als das Optimierungs-Member angegeben ist, rufen Sie die Prozedur ADMIN_CMD wie folgt auf:

  ```
  CALL SYSPROC.ADMIN_CMD('get stmm tuning member')
  ```

- Um das Optimierungs-Member zu ändern, rufen Sie die Prozedur ADMIN_CMD wie folgt auf:

  ```
  CALL SYSPROC.ADMIN_CMD('update stmm tuning member membernummer')
  ```

  Das Optimierungs-Member kann so definiert sein, dass es entweder willkürlich ausgewählt wird oder aber bei Bedarf explizit ein Optimierungs-Member festgelegt wird.

Wenn ein anderes Member für die Optimierung verwendet wird, werden einige Daten, die vom Member gesammelt wurden, das bisher die Optimierung ausgeführt hat, verworfen. Diese Daten müssen dann auf dem neuen Optimierungs-Member erneut gesammelt werden. In diesem kurzen Zeitraum, in dem die Daten neu gesammelt werden, wird die Speicheroptimierung weiterhin für das System durchgeführt. Der Optimierungsvorgang kann sich jedoch von der auf dem ursprünglich verwendeten Member durchgeführten Optimierung leicht unterscheiden.

Starten der Speicheroptimierung in einer DB2 pureScale-Umgebung

In einer DB2 pureScale-Umgebung wird die Speicheroptimierung immer dann ausgeführt, wenn die Datenbank auf mindestens einem Member aktiv ist, für das der Parameter **self_tuning_mem** auf den Wert ON gesetzt ist.

Inaktivieren der automatischen Speicheroptimierungsfunktion für ein bestimmtes Member

- Zur Inaktivierung der automatischen Speicheroptimierung für eine Teilgruppe von Datenbankmembern setzen Sie den Datenbankkonfigurationsparameter **self_tuning_mem** für diese Member auf den Wert OFF.
- Setzen Sie zur Inaktivierung der automatischen Speicheroptimierung für eine Teilgruppe von Speicherkonsumenten, die durch Konfigurationsparameter gesteuert werden, auf einem bestimmten Member den Wert des relevanten Konfigurationsparameter auf einen spezifischen Wert für dieses Member. Es wird empfohlen, die Werte der Konfigurationsparameter für die automatische Speicheroptimierungsfunktion über alle aktiven Member hinweg einheitlich zu definieren.
- Zur Inaktivierung der automatischen Speicheroptimierung für einen bestimmten Pufferpool auf einem bestimmten Member setzen Sie die Anweisung ALTER BUFFERPOOL ab, indem Sie einen Größenwert sowie das Member angeben, auf dem die automatische Speicheroptimierung inaktiviert werden soll.

Eine Anweisung ALTER BUFFERPOOL, die die Größe eines Pufferpools auf einem bestimmten Member angibt, erstellt einen Ausnahmeeintrag (bzw. aktualisiert einen vorhandenen Eintrag) für diesen Pufferpool in der Katalogsicht SYSCAT.BUFFERPOOLEXCEPTIONS. Wenn ein Ausnahmeeintrag für einen Pufferpool vorhanden ist, wird dieser Pufferpool an Operationen zur automatischen Optimierung nicht beteiligt, wenn die Standardpufferpoolgröße auf den Wert AUTOMATIC gesetzt ist.

Gehen Sie daher wie folgt vor, wenn Sie einen Ausnahmeeintrag entfernen möchten, sodass ein Pufferpool für die automatische Optimierung wieder verwendet werden kann:

1. Inaktivieren Sie die automatische Optimierung für diesen Pufferpool, indem Sie eine Anweisung ALTER BUFFERPOOL ausführen, die die Pufferpoolgröße auf einen bestimmten Wert setzt.
2. Führen Sie eine weitere Anweisung ALTER BUFFERPOOL aus, um die Größe des Pufferpools auf diesem Member auf den Standardwert zu setzen.
3. Aktivieren Sie die automatische Optimierung für diesen Pufferpool, indem Sie eine weitere Anweisung ALTER BUFFERPOOL ausführen, die die Pufferpoolgröße auf den Wert AUTOMATIC setzt.

Aktivieren des Speichers mit automatischer Leistungsoptimierung in nicht einheitlichen Umgebungen

Es wird erwartet, dass die Workloads, die auf den einzelnen Membern ausgeführt werden, einen ähnlichen Speicherbedarf aufweisen. Die Speicheranforderungen können dabei für die einzelnen Member unterschiedlich sein. Dies kann beispielsweise der Fall sein, wenn ressourcenintensive Sortiervorgänge nur auf einem Member ausgeführt werden oder wenn einige Member mit anderer Hardware und mehr verfügbarem Speicher als andere ausgestattet sind. Die automatische Speicheroptimierung kann jedoch immer noch auf einigen Membern aktiviert sein. Zur Nutzung der Vorteile der automatischen Speicheroptimierung in heterogenen Speicherumgebungen ermitteln Sie eine Gruppe von Membern, die ähnliche Speicheranforderungen haben, und aktivieren für diese Member die automatische Speicheroptimierung. Die automatische Speicheroptimierung kann auf den verbleibenden Membern inaktiviert und die Speicher manuell konfiguriert werden.

Verwenden von Speicher mit automatischer Leistungsoptimierung in Umgebungen mit partitionierten Datenbanken

Wenn die Funktion für Speicher mit automatischer Leistungsoptimierung in Umgebungen mit partitionierten Datenbanken aktiviert wird, überwacht eine einzige Datenbankpartition (die *Optimierungspartition*) die Speicherkonfiguration und gibt alle Konfigurationsänderungen an alle anderen Datenbankpartitionen weiter, um eine konsistente Konfiguration über alle beteiligten Datenbankpartitionen hinweg sicherzustellen.

Die Optimierungspartition wird nach einer Reihe von Merkmalen ausgewählt, wie zum Beispiel der Anzahl von Datenbankpartitionen in der Partitionsgruppe und der Anzahl der Pufferpools.

- Zur Ermittlung, welche Datenbankpartition zurzeit als Optimierungspartition angegeben ist, rufen Sie die Prozedur **ADMIN_CMD** wie folgt auf:

  ```
  CALL SYSPROC.ADMIN_CMD( 'get stmm tuning dbpartitionnum' )
  ```

- Zum Ändern der Optimierungspartition rufen Sie die Prozedur **ADMIN_CMD** wie folgt auf:

  ```
  CALL SYSPROC.ADMIN_CMD('update stmm tuning dbpartitionnum <partitionsnummer>')
  ```

  Die Optimierungspartition wird asynchron oder beim nächsten Start der Datenbank aktualisiert. Wenn die Speicheroptimierungsfunktion die Optimierungspartition automatisch auswählen soll, geben Sie '-1' für *partitionsnummer* ein.

Starten der Speicheroptimierungsfunktion in Umgebungen mit partitionierten Datenbanken

In einer Umgebung mit partitionierten Datenbanken wird die Speicheroptimierungsfunktion nur gestartet, wenn die Datenbank explizit mit dem Befehl **ACTIVATE DATABASE** aktiviert wird, weil die automatische Speicheroptimierung voraussetzt, dass alle Partitionen aktiv sind.

Inaktivieren der automatischen Speicheroptimierungsfunktion für eine bestimmte Datenbankpartition

- Zur Inaktivierung der automatischen Speicheroptimierung für eine Teilgruppe von Datenbankpartitionen setzen Sie den Datenbankkonfigurationsparameter **self_tuning_mem** für diese Datenbankpartitionen auf den Wert OFF.
- Zur Inaktivierung der automatischen Speicheroptimierung für eine Teilgruppe von Speicherkonsumenten, die durch Konfigurationsparameter gesteuert werden, in einer bestimmten Datenbankpartition setzen Sie den Wert des relevanten Konfigurationsparameters oder die Pufferpoolgröße auf MANUAL bzw. einen bestimmten Wert in dieser Datenbankpartition. Es wird empfohlen, die Werte der Konfigurationsparameter für die automatische Speicheroptimierungsfunktion über alle aktiven Partitionen hinweg einheitlich zu definieren.
- Zur Inaktivierung der automatischen Speicheroptimierung für einen bestimmten Pufferpool in einer bestimmten Datenbankpartition führen Sie die Anweisung ALTER BUFFERPOOL aus, indem Sie einen Größenwert sowie die Partition angeben, in der die automatische Speicheroptimierung inaktiviert werden soll.

  Eine Anweisung ALTER BUFFERPOOL, die die Größe eines Pufferpools in einer bestimmten Datenbankpartition angibt, erstellt einen Ausnahmeeintrag (bzw. aktualisiert einen vorhandenen Eintrag) für diesen Pufferpool in der Katalogsicht SYSCAT.BUFFERPOOLDBPARTITIONS. Wenn ein Ausnahmeeintrag für einen Pufferpool vorhanden ist, wird dieser Pufferpool an Operationen zur automatischen Optimierung nicht beteiligt, wenn die Standardpufferpoolgröße auf den Wert AUTOMATIC gesetzt ist. Gehen Sie daher wie folgt vor, wenn Sie einen Ausnahmeeintrag entfernen möchten, sodass ein Pufferpool für die automatische Optimierung wieder aktiviert werden kann:

  1. Inaktivieren Sie die automatische Optimierung für diesen Pufferpool, indem Sie eine Anweisung ALTER BUFFERPOOL ausführen, die die Pufferpoolgröße auf einen bestimmten Wert setzt.
  2. Führen Sie eine weitere Anweisung ALTER BUFFERPOOL aus, um die Größe des Pufferpools in dieser Datenbankpartition auf den Standardwert zu setzen.
  3. Aktivieren Sie die automatische Optimierung für diesen Pufferpool, indem Sie eine weitere Anweisung ALTER BUFFERPOOL ausführen, die die Pufferpoolgröße auf den Wert AUTOMATIC setzt.

Aktivieren des Speichers mit automatischer Leistungsoptimierung in nicht einheitlichen Umgebungen

Im Idealfall sollten Daten gleichmäßig auf alle Datenbankpartitionen verteilt und die Auslastung, die in jeder Partition ausgeführt wird, durch ähnliche Speicheranforderungen gekennzeichnet sein. Wenn die Datenverteilung ungleichmäßig ist, sodass mindestens eine Datenbankpartition erheblich mehr oder weniger Daten als andere Datenbankpartitionen enthält, sollte für solche anomalen Datenbankpartitionen die automatische Speicheroptimierung nicht aktiviert werden. Dasselbe gilt, wenn die Speicheranforderungen in den Datenbankpartitionen unterschiedlich sind. Dies kann geschehen, wenn zum Beispiel ressourcenintensive Sortiervorgänge

nur in einer Partition ausgeführt werden oder wenn einige Datenbankpartitionen mit anderer Hardware und mehr verfügbarem Speicher als andere Partitionen ausgestattet sind. Die automatische Speicheroptimierung kann dennoch in einigen Datenbankpartitionen in diesem Typ von Umgebung aktiviert werden. Zur Nutzung der Vorteile der automatischen Speicheroptimierung in Umgebungen mit ungleich verteilten Anforderungen, ermitteln Sie eine Gruppe von Datenbankpartitionen, die ähnliche Daten- und Speicheranforderungen haben, und aktivieren für diese die automatische Speicheroptimierung. Der Speicher in den übrigen Partitionen muss manuell konfiguriert werden.

Konfigurieren von Speicher und der Zwischenspeicher

Mit der Funktion zur vereinfachten Speicherkonfiguration können Sie Speicher und Zwischenspeicher konfigurieren, die für den DB2-Datenserver erforderlich sind, indem Sie die Standardeinstellung AUTOMATIC für die meisten speicherbezogenen Konfigurationsparameter verwenden. Dies ermöglicht gleichzeitig eine erhebliche Verringerung des Optimierungsaufwands.

Die vereinfachte Speicherkonfigurationsfunktion hat die folgenden Vorteile:

- Sie können einen einzigen Parameter (**instance_memory**) verwenden, um den gesamten Speicher anzugeben, den der Datenbankmanager aus seinen privaten und gemeinsam genutzten Zwischenspeichern zuordnen darf. Darüber hinaus können Sie auch den Konfigurationsparameter **appl_memory** verwenden, um die maximale Menge an Anwendungsspeicher zu steuern, die Serviceanwendungsanforderungen durch DB2-Datenbankagenten zugeordnet wird.
- Sie brauchen Parameter, die nur für den funktionalen Speicher verwendet werden, nicht manuell zu optimieren.
- Sie können den Befehl **db2mtrk** zum Überwachen der Zwischenspeicherbelegung und die Tabellenfunktion ADMIN_GET_MEM_USAGE zum Abfragen der Gesamtspeicherbelegung verwenden.
- Für die DB2-Standardkonfiguration ist viel weniger Optimierungsaufwand erforderlich. Dies ist ein Vorteil für neue Instanzen, die Sie erstellen.

In der folgenden Tabelle werden die Speicherkonfigurationsparameter aufgeführt, die standardmäßig die Einstellung AUTOMATIC haben. Diese Parameter können bei Bedarf auch dynamisch konfiguriert werden. Beachten Sie, dass sich die Bedeutung der Einstellung AUTOMATIC für die einzelnen Parameter unterscheidet. Dies wird in der ganz rechten Spalte beschrieben.

Tabelle 3. Speicherkonfigurationsparameter mit dem Standardwert AUTOMATIC

Name des Konfigurationsparameters	Beschreibung	Bedeutung der Einstellung AUTOMATIC
appl_memory	Steuert die maximale Menge an Anwendungsspeicher, die Serviceanwendungsanforderungen von DB2-Datenbankagenten zugeordnet wird.	Wenn eine durch den Parameter **instance_memory** angegebene Begrenzung festgelegt ist, lässt die Einstellung AUTOMATIC alle Anwendungsspeicheranforderungen zu, solange sich die gesamte von der Datenbankpartition zugeordnete Speicherkapazität innerhalb des durch **instance_memory** festgelegten Grenzwerts bewegt. Andernfalls lässt sie Anforderungen zu, solange Systemressourcen verfügbar sind.

Tabelle 3. Speicherkonfigurationsparameter mit dem Standardwert AUTOMATIC (Forts.)

Name des Konfigurationsparameters	Beschreibung	Bedeutung der Einstellung AUTOMATIC
applheapsz	Vor Version 9.5 bezog sich dieser Parameter auf die Größe des Anwendungsspeichers, die jeder Datenbankagent, der für eine Anwendung arbeitet, in Anspruch nehmen konnte. In Version 9.5 bezieht sich dieser Parameter auf die gesamte Anwendungsspeichergröße, die von der gesamten Anwendung verwendet werden kann. Für Umgebungen mit partitionierten Datenbanken und für Konzentrator- oder SMP-Konfigurationen bedeutet dies, dass der in früheren Releases verwendete Wert für den Parameter **applheapsz** eventuell erhöht werden muss, sofern nicht die Einstellung AUTOMATIC verwendet wird.	Mit der Einstellung AUTOMATIC kann die Größe des Anwendungszwischenspeichers nach Bedarf anwachsen. Eine Begrenzung könnte erzwungen werden, wenn eine Begrenzung durch den Parameter **appl_memory** oder **instance_memory** definiert wird.
database_memory (Vor Version 9.5 galt die Standardeinstellung AUTOMATIC nur für Windows- und AIX-Plattformen. Mit Version 9.5 gilt die Standardeinstellung AUTOMATIC für alle DB2-Serverprodukte.)	Gibt die Menge des gemeinsam genutzten Speichers an, die für den gemeinsam genutzten Speicherbereich einer Datenbank reserviert ist.	Wenn sie aktiviert ist, bestimmt die Speicheroptimierungsfunktion den Gesamtspeicherbedarf für die Datenbank und erhöht oder verringert auf der Basis des aktuellen Bedarfs der Datenbank die Speichergröße, die für den gemeinsam genutzten Datenbankspeicher zugeordnet ist.
dbheap	Legt die maximale Speicherkapazität fest, die vom Zwischenspeicher für die Datenbank verwendet wird.	Mit der Einstellung AUTOMATIC kann die Größe des Datenbankzwischenspeichers nach Bedarf anwachsen. Eine Begrenzung könnte erzwungen werden, wenn eine Begrenzung durch den Parameter **database_memory** oder **instance_memory** definiert wird.
instance_memory	Wenn ein DB2-Datenbankprodukt mit Speichernutzungsbegrenzungen verwendet oder dieser Parameter auf einen bestimmten Wert gesetzt wird, gibt dieser Parameter die maximale Speichermenge an, die einer Datenbankpartition zugeordnet werden kann.	Durch die Einstellung AUTOMATIC kann der von der gesamten Datenbankmanagerinstanz belegte Gesamtspeicher nach Bedarf anwachsen, wobei die automatische Speicheroptimierungsfunktion (STMM) sicherstellt, dass genügend Systemspeicher zur Verfügung steht, um eine Speicherüberlastung zu vermeiden. Für DB2-Datenbankprodukte mit Speichernutzungsbegrenzungen legt die Einstellung AUTOMATIC eine Grenze fest, die auf dem niedrigeren von zwei Werten basiert: einem berechneten Wert (75 - 95 % des Hauptspeichers) und dem Wert der unter der verwendeten Lizenz zulässigen Speichernutzung. Informationen dazu, wann der Wert des Parameters **instance_memory** als Grenze festgelegt wird, finden Sie in der Beschreibung dieses Parameters.

Tabelle 3. Speicherkonfigurationsparameter mit dem Standardwert AUTOMATIC (Forts.)

Name des Konfigurationsparameters	Beschreibung	Bedeutung der Einstellung AUTOMATIC
mon_heap_sz	Legt den Speicherbereich in Seiten fest, der für Daten des Datenbanksystemmonitors zugeordnet werden soll.	Mit der Einstellung AUTOMATIC kann die Größe des Monitorzwischenspeichers nach Bedarf anwachsen. Eine Begrenzung könnte erzwungen werden, wenn eine Begrenzung durch den Parameter **instance_memory** definiert wird.
stat_heap_sz	Gibt die maximale Größe des Zwischenspeichers an, der bei der Erfassung statistischer Daten mit dem Befehl **RUNSTATS** verwendet wird.	Mit der Einstellung AUTOMATIC kann die Größe des Statistikzwischenspeichers nach Bedarf anwachsen. Eine Begrenzung könnte erzwungen werden, wenn eine Begrenzung durch den Parameter **appl_memory** oder **instance_memory** definiert wird.
stmtheap	Legt die Größe des Anweisungszwischenspeichers fest, der als Arbeitsbereich für den SQL- oder XQuery-Compiler zur Kompilierung einer SQL- oder XQuery-Anweisung verwendet wird.	Mit der Einstellung AUTOMATIC kann die Größe des Anweisungszwischenspeichers nach Bedarf anwachsen. Eine Begrenzung könnte erzwungen werden, wenn eine Begrenzung durch den Parameter **appl_memory** oder **instance_memory** definiert wird.

Anmerkung: Die Verwaltungssichten DBMCFG und DBCFG rufen Informationen zu Konfigurationsparametern des Datenbankmanagers für die momentan verbundene Datenbank für alle Datenbankpartitionen ab. Für die Konfigurationsparameter **mon_heap_sz**, **stmtheap** und **stat_heap_sz** ist die Spalte DEFERRED_VALUE in diesen Sichten über Datenbankaktivierungen hinweg nicht persistent. Das heißt, dass die Ausgabe der Abfrage, wenn Sie den Befehl **get dbm cfg show detail** oder **get db cfg show detail** absetzen, aktualisierte Werte (im Speicher) anzeigt.

Der folgenden Tabelle ist zu entnehmen, ob Konfigurationsparameter beim Upgrade oder Erstellen einer Instanz und beim Upgrade oder Erstellen einer Datenbank auf den Standardwert AUTOMATIC gesetzt werden.

Tabelle 4. Konfigurationsparameter, die beim Upgrade und beim Erstellen einer Instanz oder Datenbank auf AUTOMATIC gesetzt werden

Konfigurationsparameter	Beim Upgrade oder beim Erstellen einer Instanz auf **AUTOMATIC** gesetzt	Beim Upgrade einer Datenbank auf **AUTOMATIC** gesetzt	Beim Erstellen einer Datenbank auf **AUTOMATIC** gesetzt
applheapsz[1]		X	X
dbheap		X	X
instance_memory	X		
mon_heap_sz[1]	X		
stat_heap_sz[1]		X	X
stmtheap[1]			X

Die folgenden Elemente werden für die vereinfachte Speicherkonfiguration nicht weiter unterstützt:

- Die Konfigurationsparameter **appgroup_mem_sz**, **groupheap_ratio** und **app_ctl_heap_sz**. Diese Konfigurationsparameter wurden durch den neuen Konfigurationsparameter **appl_memory** ersetzt.

- Der Parameter **-p** des Speichertrackerbefehls **db2mtrk**. Diese Option, mit der private Agentenzwischenspeicher aufgelistet werden, wurde durch den Parameter **-a** ersetzt, mit dem die gesamte Anwendungsspeicherbelegung aufgeführt wird.

Konfiguration des Agenten und des Prozessmodells

Ab Version 9.5 sind DB2-Datenbanken mit einem weniger komplexen und flexibleren Mechanismus zur Konfiguration von Prozessmodellparametern ausgestattet. Dank dieser vereinfachten Konfiguration brauchen diese Parameter nicht regelmäßig angepasst zu werden, sodass sich der Zeit- und Arbeitsaufwand für ihre Konfiguration verringert. Darüber hinaus müssen DB2-Instanzen auch nicht mehr beendet und erneut gestartet werden, um die neuen Werte in Kraft zu setzen.

Für die dynamische und automatische Agenten- und Speicherkonfiguration sind geringfügig mehr Speicherressourcen erforderlich, wenn eine Instanz aktiviert wird.

Konfiguration von Agenten, des Prozessmodells und des Speichers - Übersicht

DB2-Datenserver nutzen eine Multithreadarchitektur auf 32- und 64-Bit-Plattformen, um eine Reihe von Vorteilen, wie zum Beispiel besseren Bedienungskomfort, bessere gemeinsame Ressourcennutzung, geringeren Speicherbedarf und eine konsistente Threading-Architektur über alle Betriebssysteme hinweg zu realisieren.

In der folgenden Tabelle werden die Abschnitte zur Agenten-, Prozess- und Speicherkonfiguration nach Kategorie aufgelistet:

Tabelle 5. Übersicht über die Informationen zur Agenten-, Prozess- und Speicherkonfiguration

Kategorie	Zugehörige Themen
Allgemeine Informationen, Einschränkungen und Inkompatibilitäten	• „Konfigurieren von Speicher und der Zwischenspeicher" auf Seite 44 • „Konfiguration des Agenten und des Prozessmodells" • „Das DB2-Prozessmodell" in *Fehlerbehebung und Optimieren der Datenbankleistung* • „Konfigurieren von Datenbanken über mehrere Partitionen" auf Seite 50
Installation und Upgrade	• „Verbindungskonzentrator" in *DB2 Connect - Benutzerhandbuch* • „DB2 Connect-Optimierung" in *DB2 Connect - Benutzerhandbuch* • „Aspekte der OS/390- und zSeries SYSPLEX-Ausnutzung" in *DB2 Connect - Benutzerhandbuch* • „Platten- und Speicherbedarf" in *DB2-Server - Installation* • „Ändern von Kernelparametern (Linux)" in *DB2-Server - Installation* • „Änderungen am Verhalten des DB2-Servers" in *Upgrade auf DB2 Version 10.1*

Tabelle 5. Übersicht über die Informationen zur Agenten-, Prozess- und Speicherkonfiguration (Forts.)

Kategorie	Zugehörige Themen
Leistung	• „Verbesserungen des Verbindungskonzentrators für Clientverbindungen" in *Fehlerbehebung und Optimieren der Datenbankleistung* • „Datenbankagenten" in *Fehlerbehebung und Optimieren der Datenbankleistung* • „Verwaltung von Datenbankagenten" in *Fehlerbehebung und Optimieren der Datenbankleistung* • „Gemeinsam genutzter Speicher des Datenbankmanagers" in *Fehlerbehebung und Optimieren der Datenbankleistung* • „Hauptspeicherzuordnung in DB2" in *Fehlerbehebung und Optimieren der Datenbankleistung* • „Optimieren der Parameter für die Hauptspeicherzuordnung" in *Fehlerbehebung und Optimieren der Datenbankleistung*
Befehle, APIs, Registrierdatenbankvariablen, Funktionen und Routinen	• „db2pd - DB2-Datenbanküberwachung und -fehlerbehebung (Befehl)" in *Command Reference* • „GET DATABASE MANAGER CONFIGURATION (Befehl)" in *Command Reference* • „RESET DATABASE MANAGER CONFIGURATION (Befehl)" in *Command Reference* • „UPDATE DATABASE MANAGER CONFIGURATION (Befehl)" in *Command Reference* • „db2mtrk - Speichertracker (Befehl)" in *Command Reference* • „sqlfupd (Datenstruktur)" in *Administrative API Reference* • • „Gemeinsam genutzte Tabelle für Dateikennungen" auf Seite 52 • „Ausführen von Bibliotheksfunktionen anderer Anbieter in Prozessen im abgeschirmten Modus" auf Seite 52 • „ADMIN_GET_MEM_USAGE (Tabellenfunktion) - Gesamtspeicherbelegung für Instanz abrufen" in *Administrative Routines and Views* • „SQL- und XQuery-Begrenzungen" in *SQL Reference Volume 1* • „SYSCAT.PACKAGES (Katalogsicht)" in *SQL Reference Volume 1* • „DBMCFG (Verwaltungssicht) - Informationen zu Konfigurationsparametern des Datenbankmanagers abrufen" in *Administrative Routines and Views* • „ADMIN_CMD (Prozedur) – Verwaltungsbefehle ausführen" in *Administrative Routines and Views*

Tabelle 5. Übersicht über die Informationen zur Agenten-, Prozess- und Speicherkonfiguration (Forts.)

Kategorie	Zugehörige Themen
Konfigurations-parameter	• „Konfigurationsparameter - Zusammenfassung" auf Seite 722 • „appl_memory - Anwendungsspeicher (Konfigurationsparameter)" auf Seite 841 • „applheapsz - Zwischenspeichergröße für Anwendungen" auf Seite 842 • „database_memory - Größe des gemeinsam genutzten Datenbankspeichers" auf Seite 862 • „dbheap - Zwischenspeicher für Datenbank" auf Seite 865 • „instance_memory - Instanzspeicher" auf Seite 792 • „locklist - Maximaler Speicher für Sperrenliste" auf Seite 893 • „max_connections - Maximale Anzahl von Clientverbindungen" auf Seite 799 • „max_coordagents - Maximale Anzahl koordinierender Agenten" auf Seite 800 • „maxappls - Maximale Anzahl aktiver Anwendungen" auf Seite 911 • „mon_heap_sz - Zwischenspeichergröße für Datenbanksystemmonitor" auf Seite 805 • „num_poolagents - Agentenpoolgröße" auf Seite 809 • „stat_heap_sz - Größe des Statistikzwischenspeichers" auf Seite 955 • „stmtheap - Größe des Anweisungszwischenspeichers" auf Seite 957

Tabelle 5. Übersicht über die Informationen zur Agenten-, Prozess- und Speicherkonfiguration (Forts.)

Kategorie	Zugehörige Themen
Monitorelemente	• „Agenten und Verbindungen" in *Datenbanküberwachung - Handbuch und Referenz* • „agents_from_pool - Aus dem Pool zugeordnete Agenten" in *Datenbanküberwachung - Handbuch und Referenz* • „agents_registered - Registrierte Agenten" in *Datenbanküberwachung - Handbuch und Referenz* • „agents_registered_top - Maximale Anzahl registrierter Agenten" in *Datenbanküberwachung - Handbuch und Referenz* • „agents_stolen - Neu zugeordnete Agenten" in *Datenbanküberwachung - Handbuch und Referenz* • „appls_in_db2 - Zurzeit in der Datenbank ausgeführte Anwendungen" in *Datenbanküberwachung - Handbuch und Referenz* • „associated_agents_top - Maximale Anzahl zugeordneter Agenten" in *Datenbanküberwachung - Handbuch und Referenz* • „coord_agents_top - Maximale Anzahl koordinierender Agenten" in *Datenbanküberwachung - Handbuch und Referenz* • „local_cons - Lokale Verbindungen" in *Datenbanküberwachung - Handbuch und Referenz* • „local_cons_in_exec - Im Datenbankmanager zurzeit ausgeführte lokale Verbindungen" in *Datenbanküberwachung - Handbuch und Referenz* • „num_gw_conn_switches - Maximale Anzahl Agentenüberläufe" in *Datenbanküberwachung - Handbuch und Referenz* • „rem_cons_in - Ferne Verbindungen zum Datenbankmanager" in *Datenbanküberwachung - Handbuch und Referenz* • „rem_cons_in_exec - Im Datenbankmanager zurzeit aufgeführte ferne Verbindungen" in *Datenbanküberwachung - Handbuch und Referenz*

Konfigurieren von Datenbanken über mehrere Partitionen

Der Datenbankmanager stellt in einer Sicht alle Datenbankkonfigurationselemente über mehrere Partitionen hinweg dar. Dies bedeutet, dass Sie eine Datenbankkonfiguration über alle Datenbankpartitionen hinweg aktualisieren oder zurücksetzen können, ohne den Befehl **db2_all** für jede einzelne Datenbankpartition aufrufen zu müssen.

Sie können eine Datenbankkonfiguration über Partitionen hinweg aktualisieren, indem Sie nur eine SQL-Anweisung bzw. einen Verwaltungsbefehl von einer beliebigen Partition aus ausführen, in der sich die Datenbank befindet. Die Methode zum Aktualisieren und Zurücksetzen einer Datenbankkonfiguration gilt standardmäßig *für alle Datenbankpartitionen*.

Aus Gründen der Abwärtskompatibilität von Befehlsscripts und Anwendungen sind drei Optionen verfügbar:

- Sie können den Befehl **db2set** wie folgt verwenden, um die Registrierdatenbankvariable **DB2_UPDDBCFG_SINGLE_DBPARTITION** auf den Wert TRUE zu setzen:

```
DB2_UPDDBCFG_SINGLE_DBPARTITION=TRUE
```

Anmerkung: Die Einstellung der Registrierdatenbankvariablen hat keine Relevanz für Anforderungen mit dem Befehl **UPDATE DATABASE CONFIGURATION** oder **RESET DATABASE CONFIGURATION**, die Sie mithilfe der Prozedur ADMIN_CMD ausführen.

- Sie können den Parameter **DBPARTITIONNUM** entweder mit dem Befehl **UPDATE DATABASE CONFIGURATION** oder mit dem Befehl **RESET DATABASE CONFIGURATION** oder mit der Prozedur ADMIN_CMD verwenden. Wenn Sie zum Beispiel die Datenbankkonfigurationen in allen Datenbankpartitionen aktualisieren möchten, rufen Sie die Prozedur ADMIN_CMD wie folgt auf:

  ```
  CALL SYSPROC.ADMIN_CMD
    ('UPDATE DB CFG USING sortheap 1000')
  ```

  Zur Aktualisierung nur einer Datenbankpartition rufen Sie die Prozedur ADMIN_CMD wie folgt auf:

  ```
  CALL SYSPROC.ADMIN_CMD
    ('UPDATE DB CFG DBPARTITIONNUM 10 USING sortheap 1000')
  ```

- Sie können den Parameter **DBPARTITIONNUM** mit der API db2CfgSet verwenden. Die Markierungen (Flags) in der Struktur **db2Cfg** geben an, ob der Wert für die Datenbankkonfiguration auf nur eine Datenbankpartition angewendet werden soll. Wenn Sie eine Markierung setzen, müssen Sie auch den Wert für **DBPARTITIONNUM** angeben. Beispiel:

  ```
  #define db2CfgSingleDbpartition          256
  ```

  Wenn Sie den Wert für `db2CfgSingleDbpartition` nicht definieren, gilt der Wert für die Datenbankkonfiguration für alle Datenbankpartitionen, sofern Sie nicht die Registrierdatenbankvariable **DB2_UPDDBCFG_SINGLE_DBPARTITION** auf den Wert `TRUE` setzen oder für die API 'db2CfgSet', die die Konfigurationsparameter des Datenbankmanagers oder der Datenbank einstellt, das Feld *versionNumber* auf einen beliebigen Wert setzen, der kleiner als die Versionsnummer für Version 9.5 ist.

Bei einem Upgrade der Datenbanken auf Version 9.7 behalten vorhandene Datenbankkonfigurationsparameter im Allgemeinen ihre Werte nach dem Upgrade bei. Es werden jedoch neue Parameter mit Standardwerten hinzugefügt, und einige vorhandene Parameter werden auf ihre neuen Standardwerte der Version 9.7 gesetzt. Detaillierte Informationen zu den Änderungen an vorhandenen Datenbankkonfigurationsparametern enthält der Abschnitt zu Änderungen im Verhalten des DB2-Servers im Handbuch *Upgrade auf DB2 Version 10.1*. Alle nachfolgenden Anforderungen zum Aktualisieren oder Zurücksetzen der Datenbankkonfiguration für die Datenbanken, für die das Upgrade erfolgt ist, werden standardmäßig auf alle Datenbankpartitionen angewendet.

Für vorhandene Aktualisierungs- oder Zurücksetzbefehlsscripts gelten dieselben zuvor genannten Regeln für alle Datenbankpartitionen. Sie können Ihre Scripts ändern, um die Option **DBPARTITIONNUM** des Befehls **UPDATE DATABASE CONFIGURATION** oder **RESET DATABASE CONFIGURATION** einzufügen, oder Sie können die Registrierdatenbankvariable **DB2_UPDDBCFG_SINGLE_DBPARTITION** definieren.

Für vorhandene Anwendungen, die die API **db2CfgSet** aufrufen, müssen Sie nach den Anweisungen für Version 9.5 oder eine spätere Version verfahren. Wenn Sie das Verhalten vor Version 9.5 wünschen, können Sie die Registrierdatenbankvariable **DB2_UPDDBCFG_SINGLE_DBPARTITION** definieren oder Ihre Anwendungen in der Weise ändern, dass sie die API mit der Versionsnummer von Version 9.5 (oder einer späteren Version), einschließlich der neuen Markierung (Flag)

db2CfgSingleDbpartition und des neuen Felds **dbpartitionnum**, aufrufen, um Datenbankkonfigurationen für eine bestimmte Datenbankpartition zu aktualisieren bzw. zurückzusetzen.

Anmerkung: Wenn Sie feststellen, dass Datenbankkonfigurationswerte inkonsistent sind, können Sie jede Datenbankpartition einzeln aktualisieren oder zurücksetzen.

Gemeinsam genutzte Tabelle für Dateikennungen

Der mit Threads arbeitende Datenbankmanager verwaltet nur eine gemeinsam genutzte Dateikennungstabelle für jede Datenbank und alle Agenten, die für die einzelnen Datenbanken aktiv sind, sodass E/A-Anforderungen, die sich auf dieselbe Datei beziehen, kein erneutes Öffnen und Schließen der Datei erfordern.

Vor Version 9.5 wurde die Dateikennungstabelle von jedem DB2-Agenten separat verwaltet, und die Größe der Dateikennungstabelle pro Agent wurde über den Konfigurationsparameter **maxfilop** gesteuert. Ab Version 9.5 verwaltet der Datenbankmanager eine einzige gemeinsam genutzte Tabelle für Dateikennungen für die gesamte Datenbank, sodass dieselbe Dateikennung von allen Agenten, die für dieselbe Datenbankdatei aktiv sind, gemeinsam genutzt werden kann. Infolgedessen wird der Konfigurationsparameter **maxfilop** zum Steuern der Größe der gemeinsam genutzten Dateikennungstabelle verwendet.

Aufgrund dieser Änderung hat der Konfigurationsparameter **maxfilop** ab Version 9.5 einen anderen Standardwert sowie einen neuen Mindest- und Höchstwert. Beim Upgrade einer Datenbank wird der Konfigurationsparameter **maxfilop** automatisch auf diesen Standardwert gesetzt, wenn Sie ein Upgrade von einem Release vor Version 9.5 durchführen.

Ausführen von Bibliotheksfunktionen anderer Anbieter in Prozessen im abgeschirmten Modus

Der Datenbankmanager unterstützt Bibliotheksfunktionen anderer Anbieter in Prozessen des abgeschirmten Modus, die solche Aufgaben ausführen, wie Datenkomprimierung, TSM-Backups und Protokolldatenarchivierung.

Informationen zu diesem Vorgang

Vor Version 9.5 wurden Bibliotheksfunktionen, Dienstprogramme oder Routinen anderer Anbieter innerhalb von Agentenprozessen ausgeführt. Da ab Version 9.5 der DB2-Datenbankmanager selbst eine Multithreading-Anwendung ist, können Bibliotheksfunktionen anderer Anbieter, die nicht mehr threadsicher sind, Speicher- oder Stackdatenverluste oder, noch schwer wiegender, Datenbeschädigungen in DB2-Datenbanken verursachen. Aus diesem Grund wird für jeden Aufruf eines Dienstprogramms eines anderen Anbieters ein neuer Prozess im abgeschirmten Modus erstellt, und die Bibliotheksfunktionen oder Routinen des Anbieters werden in diesem Prozess im abgeschirmten Modus ausgeführt. Dies wird nicht zu wesentlichen Leistungseinbußen führen.

Anmerkung: Die Funktionalität des abgeschirmten Modus ist für Windows-Plattformen nicht verfügbar.

Dynamischer Speicher

Der dynamische Speicher vereinfacht die Speicherverwaltung für Tabellenbereiche. Sie können Speichergruppen erstellen, die aus Pfaden bestehen, in denen der Datenbankmanager Ihre Daten speichert. Anschließend verwaltet der Datenbankmanager die Container und die Speicherzuordnung für die Tabellenbereiche, je nach-

dem, wie sie von Ihnen erstellt und mit Daten gefüllt werden. Beim Erstellen der Datenbank können Sie die Pfade für die Standardspeichergruppe angeben.

Datenbanken arbeiten standardmäßig mit dynamischem Speicher

Die Funktionalität des dynamischen Speichers kann zur Vereinfachung des Speichermanagements dienen. Anstatt den Speicher auf Tabellenbereichsebene über explizite Containerdefinitionen zu verwalten, erfolgt das Speichermanagement auf Datenbankebene und die Verantwortung für das Erstellen, Erweitern und Hinzufügen von Containern wird vom Datenbankmanager übernommen.

Anmerkung: Obwohl Sie beim Erstellen einer Datenbank die Klausel AUTOMATIC STORAGE NO angeben können, ist die Klausel AUTOMATIC STORAGE veraltet und wird in einem zukünftigen Release möglicherweise entfernt.
Alle Datenbanken werden standardmäßig mit dynamischem Speicher erstellt. Wenn bei der Erstellung einer Datenbank jedoch die Klausel AUTOMATIC STORAGE NO angegeben wurde, kann die betreffende Datenbank keine im dynamischen Speicher verwalteten Tabellenbereiche verwenden.

Beim Erstellen einer Datenbank wird standardmäßig eine Standardspeichergruppe automatisch erstellt. Für diese Gruppe können Sie zu Anfang einen oder mehrere Speicherpfade erstellen. Wenn die Datenbank an Volumen zunimmt, erstellt der Datenbankmanager Container in diesen Speicherpfaden und erweitert diese bzw. erstellt neue Container je nach Bedarf automatisch. Die Liste der Speicherpfade kann in der Verwaltungssicht ADMIN_GET_STORAGE_PATHS angezeigt werden.

Wenn eine Datenbank nicht über Speichergruppen verfügt, können Sie mit der Anweisung CREATE STOGROUP eine Speichergruppe erstellen. Die neu erstellte Speichergruppe ist die Standardspeichergruppe, d. h. alle vom dynamischen Speicher verwalteten neuen Tabellenbereiche werden über diese Speichergruppe zur Datenbank hinzugefügt. Die Standardspeichergruppe kann mit der Klausel SET AS DEFAULT in der Anweisung CREATE STOGROUP oder mit der Anweisung ALTER STOGROUP geändert werden.

Wichtig:

- Durch das Hinzufügen von Speicherpfaden werden bereits vorhandene Tabellenbereiche ohne dynamischen Speicher nicht zur Verwendung von dynamischem Speicher konvertiert. Sie können DMS-Tabellenbereiche zur Verwendung von dynamischem Speicher konvertieren. SMS-Tabellenbereiche können nicht zur Verwendung von dynamischem Speicher konvertiert werden. Weitere Informationen finden Sie in „Konvertieren von Tabellenbereichen zur Verwendung von dynamischem Speicher" auf Seite 190.
- Eine Datenbank, für die Speichergruppen erstellt wurden, verfügt über mindestens eine Speichergruppe. Die letzte verbleibende Speichergruppe im Datenbankmanager kann nicht entfernt werden.
- Um eine vorhersehbare Leistung zu gewährleisten, sollten die zu einer Speichergruppe hinzugefügten Speicherpfade ähnliche Datenträgermerkmale aufweisen.

Datenkomprimierung

Sie können den für Ihre Daten erforderlichen Speicherplatz durch die Verwendung von Komprimierungsfunktionen verringern, die in DB2 for Linux, UNIX and Windows integriert sind und Ihre Tabellen, Ihre Indizes und sogar Ihre Backup-Images verkleinern.

Tabellen und Indizes enthalten häufig wiederkehrende Informationen. Diese Wiederholungen können von einzelnen oder kombinierten Spaltenwerten über gemeinsame Präfixe für Spaltenwerte bis hin zu wiederholten Mustern in XML-Daten reichen. Es steht eine Reihe von Komprimierungsfunktionen zur Verfügung, mit denen Sie den zum Speichern Ihrer Tabellen und Indizes erforderlichen Speicherplatz verringern können. Daneben sind Funktionen verfügbar, mit deren Hilfe Sie die Einsparungen ermitteln können, die sich durch die Komprimierung erzielen lassen.

Darüber hinaus können Sie durch eine Backupkomprimierung die Größe Ihrer Backups verringern.[1]

Die meisten Editionen von DB2 V9.7 enthalten die folgenden Komprimierungsfunktionen:

- Wertkomprimierung
- Backupkomprimierung

Mit der Lizenz für das DB2 Storage Optimization Feature sind die folgenden zusätzlichen Komprimierungsfunktionen verfügbar:

- Zeilenkomprimierung, einschließlich Komprimierung für XML-Speicherobjekte
- Komprimierung temporärer Tabellen
- Indexkomprimierung

Automatische Statistikerfassung

Das DB2-Optimierungsprogramm verwendet Katalogstatistiken, um den effizientesten Zugriffsplan für eine Abfrage zu ermitteln. Tabellen- oder Indexstatistiken, die nicht auf dem neuesten Stand oder unvollständig sind, könnten dazu führen, dass das Optimierungsprogramm einen nicht optimalen Plan auswählt, sodass sich die Abfrageausführung verlangsamt. Allerdings ist die Entscheidung, welche Statistiken für eine gegebene Auslastung zu erfassen sind, recht komplex und die Pflege aktueller Statistiken zeitaufwendig.

Mithilfe der automatischen Statistikerfassung, die Teil der Funktion zur automatischen Tabellenverwaltung von DB2 ist, können Sie den Datenbankmanager bestimmen lassen, ob Datenbankstatistiken aktualisiert werden müssen. Die automatische Statistikerfassung kann *synchron* zur Anwendungskompilierung durch die Verwendung der Echtzeitstatistikfunktion (RTS, Real-Time Statistics) stattfinden. Alternativ kann der Befehl **RUNSTATS** für eine *asynchrone* Erfassung zur Ausführung im Hintergrund aktiviert werden. Obwohl die Statistikerfassung im Hintergrund aktiviert werden kann, während die Echtzeitstatistikerfassung inaktiviert ist, muss die Statistikerfassung im Hintergrund aktiviert sein, damit eine Echtzeitstatistikerfassung erfolgen kann. Die automatische Statistikerfassung im Hintergrund (**auto_runstats**) und die automatische Echtzeitstatistikerfassung (**auto_stmt_stats**) werden standardmäßig aktiviert, wenn eine Datenbank erstellt wird.

Seit DB2 Version 9 können Sie den Konfigurationsadvisor dazu verwenden, die Anfangskonfiguration für neue Datenbanken zu bestimmen (einschließlich der passenden Einstellung des Datenbankkonfigurationsparameters **auto_stmt_stats**).

Ab IBM Data Studio Version 3.1 kann der Taskassistent für Folgendes verwendet werden: Konfigurieren der automatischen Statistikerfassung. Taskassistenten führen

1. Weitere Informationen finden Sie unter „Backupkomprimierung" in *Datenrecovery und hohe Verfügbarkeit - Handbuch und Referenz*.

durch den Prozess der Definition von Optionen, der Prüfung automatisch generierter Befehle für die jeweilige Task und der Ausführung dieser Befehle. Weitere Einzelheiten finden Sie in Verwalten von Datenbanken mit Taskassistenten.

Asynchron und in Echtzeit ausgeführte Statistikerfassung

Wenn die Echtzeitstatistikerfassung aktiviert ist, können Statistikdaten mithilfe bestimmter Metadaten konstruiert werden. *Konstruieren* bedeutet in diesem Fall, dass Statistiken abgeleitet oder generiert und nicht im Rahmen der normalen **RUNSTATS**-Aktivitäten erfasst werden. Zum Beispiel lässt sich die Anzahl von Zeilen in einer Tabelle aus der Kenntnis der Anzahl von Seiten in der Tabelle, der Seitengröße und der durchschnittlichen Zeilenbreite ableiten. In einigen Fällen werden die Statistiken nicht abgeleitet, sondern durch den Index- und den Datenmanager gepflegt, sodass sie direkt im Katalog gespeichert werden können. Zum Beispiel verwaltet der Indexmanager einen Zähler für die Blattseiten und Stufen in jedem Index.

Das Abfrageoptimierungsprogramm bestimmt, wie Statistiken erfasst werden. Dies geschieht auf der Basis der Anforderungen der Abfrage und des Umfangs der Tabellenaktualisierungsaktivitäten (d. h. der Anzahl der Aktualisierungs-, Einfüge- und Löschoperationen).

Die Echtzeitstatistikerfassung stellt zeitgerechtere und präzisere Statistikdaten zur Verfügung. Präzise Statistikdaten können bessere Abfrageausführungspläne und eine höhere Leistung zur Folge haben. Unabhängig davon, ob die Echtzeitstatistikerfassung aktiviert ist, erfolgt die asynchrone Statistikerfassung in Intervallen von je zwei Stunden. Dies ist möglicherweise nicht häufig genug, um präzise Statistiken für bestimmte Anwendungen bereitzustellen.

Die Echtzeitstatistikerfassung leitet in folgenden Fällen auch asynchrone Erfassungsanforderungen ein:

- Wenn das Tabellenaktivitätsvolumen nicht ausreicht, um eine synchrone Erfassung zu erfordern, jedoch ausreicht, um eine asynchrone Erfassung zu erfordern.
- Wenn die synchrone Statistikerfassung mit Stichproben gearbeitet hat, weil die Tabelle sehr groß war.
- Wenn die synchronen Statistikdaten konstruiert wurden.
- Wenn die synchrone Statistikerfassung fehlgeschlagen ist, weil die Erfassungszeit überschritten wurde.

Es können höchstens zwei asynchrone Anforderungen gleichzeitig verarbeitet werden, jedoch nur für verschiedene Tabellen. Die eine Anforderung muss durch die Echtzeitstatistikerfassung, die andere durch die Überprüfung der asynchronen Statistikerfassung eingeleitet worden sein.

Die Leistungsbeeinträchtigung durch die automatische Statistikerfassung wird auf mehrere Arten minimiert:

- Die asynchrone Statistikerfassung erfolgt durch eine gedrosselte Ausführung des Dienstprogramms **RUNSTATS**. Die Drosselung begrenzt je nach aktuellen Datenbankaktivitäten die Ressourcenkapazität, die vom Dienstprogramm **RUNSTATS** beansprucht wird: Wenn die Datenbankaktivitäten zunehmen, läuft das Dienstprogramm **RUNSTATS** langsamer, sodass sein Ressourcenbedarf sinkt.
- Die synchrone Statistikerfassung ist auf fünf Sekunden pro Abfrage begrenzt. Dieser Wert kann durch die RTS-Optimierungsrichtlinie gesteuert werden. Wenn die synchrone Erfassung das Zeitlimit überschreitet, wird eine Anforderung zur asynchronen Erfassung übergeben.

- Die synchrone Statistikerfassung speichert die Statistiken nicht im Systemkatalog. Stattdessen werden die Statistiken in einem Statistikcache gespeichert und später durch eine asynchrone Operation im Systemkatalog gespeichert. Durch diese Abfolge werden Systemaufwand und mögliche Sperrenkonflikte vermieden, die beim Aktualisieren des Systemkatalogs auftreten können. Statistiken im Statistikcache sind für nachfolgende SQL-Kompilierungsanforderungen verfügbar.
- Pro Tabelle findet nur eine synchrone Statistikerfassungsoperation statt. Andere Agenten, die eine synchrone Statistikerfassung erfordern, konstruieren Statistiken, sofern möglich, und setzen die Anweisungskompilierung fort. Dieses Verhalten wird auch in einer Umgebung mit partitionierten Datenbanken realisiert, in der Agenten in verschiedenen Datenbankpartitionen möglicherweise synchrone Statistiken benötigen.
- Sie können den Typ der Statistiken, die erfasst werden, anpassen, indem Sie die Statistikprofilerstellung aktivieren, sodass anhand von Informationen zu früheren Datenbankaktivitäten bestimmt wird, welche Statistiken für die Auslastung der Datenbank erforderlich sind. Alternativ können Sie auch ein eigenes Statistikprofil für eine bestimmte Tabelle erstellen.
- Nur Tabellen mit fehlenden Statistiken oder hohen Graden an Aktivität (gemessen an der Anzahl der Aktualisierungs-, Einfüge- und Löschoperationen) werden für die Statistikerfassung in Betracht gezogen. Auch wenn die Tabelle die Bedingungen für die Statistikerfassung erfüllt, werden die synchronen Statistiken nicht erfasst, sofern sie nicht für die Abfrageoptimierung erforderlich sind. In einigen Fällen kann das Abfrageoptimierungsprogramm einen Plan ohne Statistiken auswählen.
- Bei der asynchronen Statistikerfassungsprüfung werden für große Tabellen (Tabellen mit mehr als 4.000 Seiten) Stichproben erstellt, um festzustellen, ob die Statistiken durch intensive Tabellenaktivitäten geändert wurden. Statistiken für solch große Tabellen werden nur erfasst, wenn dies gerechtfertigt ist.
- Bei der asynchronen Statistikerfassung wird das Dienstprogramm **RUNSTATS** automatisch zur Ausführung während des Onlineverwaltungsfensters terminiert, das in Ihrer Verwaltungsrichtlinie angegeben ist. Diese Richtlinie gibt außerdem die Gruppe von Tabellen an, die zum Umfang der automatischen Statistikerfassung gehören, was zusätzlich eine unnötige Ressourcennutzung minimiert.
- Die synchrone Statistikerfassung und die Konstruktion von Statistikdaten richten sich nicht nach dem Onlineverwaltungsfensters, das in Ihrer Verwaltungsrichtlinie angegeben ist, weil synchrone Anforderungen sofort ausgeführt werden müssen und nur über eine begrenzte Erfassungszeit verfügen. Die synchrone Statistikerfassung und die Konstruktion von Statistikdaten richten sich nach der Richtlinie, die die Gruppe von Tabellen angibt, die zum Umfang der automatischen Statistikerfassung gehören.
- Während der Ausführung der automatischen Statistikerfassung bleiben die betroffenen Tabellen weiterhin für reguläre Datenbankaktivitäten (Aktualisierungs-, Einfüge- und Löschoperationen) verfügbar.
- Echtzeitstatistiken (synchrone oder konstruierte) werden für Kurznamen nicht erfasst. Zur Aktualisierung von Statistiken für Kurznamen im Systemkatalog (für die synchrone Statistikerfassung) rufen Sie die Prozedur SYSPROC.NNSTAT auf. Für die asynchrone Statistikerfassung ruft DB2 Database for Linux, UNIX and Windows automatisch die Prozedur SYSPROC.NNSAT auf, um die Statistiken für Kurznamen im Systemkatalog zu aktualisieren.
- Echtzeitstatistiken (synchrone oder konstruierte) werden für Statistiksichten nicht erfasst.

- Für deklarierte temporäre Tabellen (DGTTs) kann nur eine Echtzeitstatistik erfasst werden.

Obwohl die Echtzeitstatistikerfassung mit dem Ziel entwickelt wurde, den Aufwand für die Statistikerfassung zu minimieren, sollten Sie sie zunächst in einer Testumgebung erproben, um sicherzustellen, dass es zu keinen Leistungseinbußen kommt. In einigen OLTP-Szenarios (OLTP - Onlinetransaktionsverarbeitung) sind solche Leistungseinbußen möglich, insbesondere wenn die Ausführungsdauer einer Abfrage einer oberen Begrenzung unterliegt.

Die synchrone Echtzeitstatistikerfassung wird für reguläre Tabellen, MQTs (Materialized Query Tables) und globale temporäre Tabellen ausgeführt. Asynchrone Statistiken werden für globale temporäre Tabellen nicht erfasst.

Die automatische Statistikerfassung (synchron oder asynchron) wird für folgende Elemente nicht ausgeführt:

- Tabellen, die als flüchtig markiert sind (Tabellen, deren Feld VOLATILE in der Katalogsicht SYSCAT.TABLES definiert ist)
- Erstellte temporäre Tabellen (CGTTs)
- Tabellen, deren Statistiken durch UPDATE-Anweisungen direkt in den SYSSTAT-Katalogsichten manuell aktualisiert wurden

  Wenn Sie Tabellenstatistikdaten manuell modifizieren, geht der Datenbankmanager davon aus, dass nun Sie für die Pflege dieser Statistikdaten verantwortlich sind. Um den Datenbankmanager dazu zu veranlassen, die Statistiken für eine Tabelle, deren Statistiken manuell aktualisiert wurden, wieder zu pflegen, führen Sie den Befehl **RUNSTATS** zur Erfassung von Statistikdaten aus oder geben Sie die Statistikerfassung bei der Verwendung des Befehls **LOAD** an. Tabellen, die vor Version 9.5 erstellt wurden und deren Statistikdaten vor dem Upgrade manuell aktualisiert wurden, sind davon nicht betroffen. Die Statistikdaten dieser Tabellen werden vom Datenbankmanager weiterhin automatisch gepflegt, bis sie manuell aktualisiert werden.

Für folgende Elemente erfolgt keine Konstruktion von Statistikdaten:

- Statistiksichten
- Tabellen, deren Statistiken durch UPDATE-Anweisungen direkt in den SYSSTAT-Katalogsichten manuell aktualisiert wurden. Wenn die Echtzeitstatistikerfassung nicht aktiviert ist, werden dennoch einige Statistiken für Tabellen konstruiert, deren Statistiken manuell aktualisiert wurden.

In einer Umgebung mit partitionierten Datenbanken werden die Statistikdaten in nur einer Datenbankpartition erfasst und extrapoliert. Der Datenbankmanager erfasst Statistikdaten (sowohl synchrone als auch asynchrone) stets in der ersten Datenbankpartition der Datenbankpartitionsgruppe.

Aktivitäten zur Echtzeitstatistikerfassung finden nicht vor Ablauf der ersten fünf Minuten nach der Datenbankaktivierung statt.

Sowohl für statisches als auch für dynamisches SQL wird eine Verarbeitung der Echtzeitstatistiken ausgeführt.

Eine Tabelle, die entweder durch die Anweisung TRUNCATE oder den Befehl **IMPORT** abgeschnitten wurde, wird automatisch als Tabelle mit Statistiken erkannt, die nicht auf dem neuesten Stand sind.

Eine automatische Statistikerfassung (synchron und asynchron) macht im Cache gespeicherte dynamische Anweisungen, die auf Tabellen verweisen, für die Statistiken erfasst wurden, ungültig. Dies geschieht, damit im Cache gespeicherte dynamische Anweisungen mit den aktuellsten Statistiken reoptimiert werden können.

Asynchrone automatische Statistikerfassungsoperationen können unterbrochen werden, wenn die Datenbank inaktiviert wird. Wenn die Datenbank nicht explizit mithilfe des Befehls **ACTIVATE DATABASE** oder der API aktiviert wurde, wird die Datenbank inaktiviert, wenn der letzte Benutzer die Verbindung zur Datenbank trennt. Werden Operationen unterbrochen, werden möglicherweise Fehlernachrichten in der DB2-Diagnoseprotokolldatei aufgezeichnet. Aktivieren Sie die Datenbank explizit, um die Unterbrechung asynchroner automatischer Statistikerfassungsoperationen zu vermeiden.

Echtzeitstatistiken und EXPLAIN-Verarbeitung

Es erfolgt keine Echtzeitverarbeitung für eine Abfrage, die durch die EXPLAIN-Funktion nur bearbeitet, jedoch nicht ausgeführt wird. In der folgenden Tabelle sind die Verhaltensweisen unter verschiedenen Werten des Sonderregisters CURRENT EXPLAIN MODE zusammengefasst.

Tabelle 6. Echtzeitstatistikerfassung als Funktion des Werts des Sonderregisters CURRENT EXPLAIN MODE

Wert in CURRENT EXPLAIN MODE	Echtzeitstatistikerfassung in Betracht gezogen?
YES	Ja
EXPLAIN	Nein
NO	Ja
REOPT	Ja
RECOMMEND INDEXES	Nein
EVALUATE INDEXES	Nein

Automatische Statistikerfassung und Statistikcache

In DB2 Version 9.5 wurde ein Statistikcache eingeführt, um synchron erfasste Statistikdaten für alle Abfragen verfügbar zu machen. Dieser Cache ist Teil des Katalogcache. In einer Umgebung mit partitionierten Datenbanken befindet sich der Statistikcache nur in der Katalogdatenbankpartition, auch wenn jede Datenbankpartition über einen Katalogcache verfügt. Wenn die Echtzeitstatistikerfassung aktiviert ist, sind die Anforderungen an den Katalogcache höher. Ziehen Sie in Betracht, den Wert des Datenbankkonfigurationsparameters **catalogcache_sz** zu optimieren, wenn die Echtzeitstatistikerfassung aktiviert ist.

Seit DB2 Version 9 können Sie den Konfigurationsadvisor dazu verwenden, die Anfangskonfiguration für neue Datenbanken zu bestimmen. Der Konfigurationsadvisor empfiehlt, den Datenbankkonfigurationsparameter **auto_stmt_stats** auf den Wert ON zu setzen.

Automatische Statistikerfassung und Statistikprofile

Synchrone und asynchrone Statistiken werden entsprechend einem Statistikprofil erfasst, das für eine Tabelle in Kraft ist. Von dieser Regel gibt es folgende Ausnahmen:

- Zur Minimierung des Aufwands für die synchrone Statistikerfassung kann der Datenbankmanager Statistikdaten durch Stichprobenentnahme erfassen. In diesem Fall können die Stichprobenrate und die Methode von denen abweichen, die im Statistikprofil angegeben sind.
- Es ist möglich, dass die synchrone Statistikerfassung Statistiken konstruiert, obwohl es vielleicht nicht möglich ist, alle im Statistikprofil angegebenen Statistiken zu konstruieren. Zum Beispiel können Spaltenstatistiken wie COLCARD, HIGH2KEY und LOW2KEY nicht konstruiert werden, wenn die Spalte nicht an führender Position in einem Index enthalten ist.

Wenn die synchrone Statistikerfassung nicht alle Statistiken erfassen kann, die im Statistikprofil angegeben sind, wird eine Anforderung zur asynchronen Statistikerfassung übergeben.

Aktivieren der automatischen Statistikerfassung

Über genaue und vollständige Datenbankstatistiken zu verfügen, ist von kritischer Bedeutung für einen effizienten Datenzugriff und eine optimale Auslastungsleistung. Verwenden Sie die Funktion zur automatischen Statistikerfassung der Funktionalität zur automatischen Tabellenverwaltung, um relevante Datenbankstatistiken zu aktualisieren und zu pflegen.

Informationen zu diesem Vorgang

In Umgebungen, in denen eine einzelne Datenbankpartition auf einem Einzelprozessor betrieben wird, können Sie diese Funktionalität erweitern, indem Sie Abfragedaten sammeln und Statistikprofile generieren, die den DB2-Server bei der automatischen Erfassung der exakten Gruppe der für Ihre Auslastung erforderlichen Statistiken unterstützen. Diese Option ist in Umgebungen mit partitionierten Datenbanken, in bestimmten Umgebungen mit föderierten Datenbanken oder in Umgebungen mit aktivierter partitionsinterner Parallelität nicht verfügbar.

Für die Aktivierung der automatischen Statistikerfassung müssen Sie zuerst die Datenbank entsprechend konfigurieren, indem Sie die Datenbankkonfigurationsparameter **auto_maint** und **auto_tbl_maint** auf den Wert ON setzen.

Vorgehensweise

Wenn die Datenbankkonfigurationsparameter **auto_maint** und **auto_tbl_maint** auf den Wert ON gesetzt wurden, haben Sie folgende Optionen:

- Zur Aktivierung der Statistikerfassung im Hintergrund setzen Sie den Datenbankkonfigurationsparameter **auto_runstats** auf den Wert ON.
- Um die Hintergrundstatistikerfassung für Statistiksichten zu aktivieren, setzen Sie die Datenbankkonfigurationsparameter **auto_stats_views** und **auto_runstats** beide auf ON.
- Wenn Sie die Statistikerfassung im Hintergrund aktivieren wollen, um für große Tabellen und Statistiksichten die automatische Stichprobenentnahme zu verwenden, setzen Sie auch den Datenbankkonfigurationsparameter **auto_sampling** auf den Wert ON. Verwenden Sie diese Einstellung zusätzlich zu **auto_runstats** (nur Tabellen) oder zu **auto_runstats** und **auto_stats_views** (Tabellen und Statistiksichten).
- Zur Aktivierung der Statistikerfassung in Echtzeit setzen Sie sowohl den Datenbankkonfigurationsparameter **auto_stmt_stats** als auch den Datenbankkonfigurationsparameter **auto_runstats** auf den Wert ON.

Konfigurationsadvisor

Mithilfe des Konfigurationsadvisors können Sie Empfehlungen für die Anfangswerte der Pufferpoolgröße, der Datenbankkonfigurationsparameter und der Konfigurationsparameter des Datenbankmanagers abrufen.

Zur Verwendung des Konfigurationsadvisors geben Sie den Befehl **AUTOCONFIGURE** für eine vorhandene Datenbank oder beim Erstellen einer Datenbank die Option **AUTOCONFIGURE** im Befehl **CREATE DATABASE** an. Zur Konfiguration Ihrer Datenbank müssen Sie über die Berechtigung SYSADM, SYSCTRL oder SYSMAINT verfügen.

Sie können die empfohlenen Werte anzeigen oder sie durch Angeben des Parameters APPLY in den Befehlen **CREATE DATABASE** und **AUTOCONFIGURE** anwenden. Die Empfehlungen basieren auf Ihrer Eingabe sowie auf Systeminformationen, die von der Advisorfunktion erfasst werden.

Die vom Konfigurationsadvisor vorgeschlagenen Werte sind nur für eine Datenbank pro Instanz relevant. Wenn Sie diese Advisorfunktion für mehrere Datenbanken verwenden möchten, muss jede Datenbank zu einer separaten Instanz gehören.

Optimieren von Konfigurationsparametern mit dem Konfigurationsadvisor

Der Konfigurationsadvisor hilft Ihnen bei der Optimierung der Leistung und der ausgewogenen Verteilung des Speicherbedarfs für eine einzelne Datenbank pro Instanz, indem er Konfigurationsparameter zur Änderung vorschlägt und geeignete Werte für sie empfiehlt. Der Konfigurationsadvisor wird automatisch ausgeführt, wenn Sie eine Datenbank erstellen.

Informationen zu diesem Vorgang

Zur Inaktivierung dieser Funktion oder zur expliziten Aktivierung dieser Funktion verwenden Sie den Befehl **db2set** wie folgt, bevor Sie eine Datenbank erstellen:

```
db2set DB2_ENABLE_AUTOCONFIG_DEFAULT=NO
db2set DB2_ENABLE_AUTOCONFIG_DEFAULT=YES
```

Zum Definieren von Werten für verschiedene Konfigurationsparameter und zum Bestimmen des Anwendungsbereichs dieser Parameter verwenden Sie den Befehl **AUTOCONFIGURE**, indem Sie eine der folgenden Optionen angeben:

- NONE: Bedeutet, dass keiner der Werte angewendet wird.
- DB ONLY: Bedeutet, dass nur Werte für die Datenbankkonfiguration und Pufferpools angewendet werden.
- DB AND DBM: Bedeutet, dass alle Parameter und ihre Werte angewendet werden.

Anmerkung: Auch wenn der Konfigurationsadvisor bei der Ausführung des Befehls **CREATE DATABASE** automatisch aktiviert ist, können Sie dennoch die Optionen für den Befehl **AUTOCONFIGURE** angeben. Wenn Sie den Konfigurationsadvisor bei der Ausführung des Befehls **CREATE DATABASE** nicht aktiviert haben, können Sie den Konfigurationsadvisor anschließend manuell ausführen.

Generieren von Konfigurationsempfehlungen

Der Konfigurationsadvisor wird automatisch ausgeführt, wenn Sie eine Datenbank erstellen. Sie können den Konfigurationsadvisor auch ausführen, indem Sie den Befehl **AUTOCONFIGURE** in den Befehlszeilenprozessor (CLP) eingeben oder die API db2AutoConfig aufrufen.

Vorgehensweise

Zur Anforderung von Konfigurationsempfehlungen über den Befehlszeilenprozessor geben Sie den folgenden Befehl ein:

```
AUTOCONFIGURE
     USING eingabeschlüsselwort parameterwert
     APPLY wert
```

Beispiel

Das folgende Beispiel zeigt einen Befehl **AUTOCONFIGURE**, mit dem Konfigurationsempfehlungen auf der Basis von Eingaben über die Nutzung der Datenbank angefordert werden, ohne die Empfehlungen jedoch anzuwenden:

```
    DB2 AUTOCONFIGURE USING
      MEM_PERCENT 60
      WORKLOAD_TYPE MIXED
      NUM_STMTS 500
      ADMIN_PRIORITY BOTH
      IS_POPULATED YES
      NUM_LOCAL_APPS 0
      NUM_REMOTE_APPS 20
      ISOLATION RR
      BP_RESIZEABLE YES
APPLY NONE
```

Beispiel: Anfordern von Konfigurationsempfehlungen mit dem Konfigurationsadvisor

Dieses Szenario veranschaulicht, wie der Konfigurationsadvisor über die Befehlszeile ausgeführt wird, um Empfehlungen zu generieren, und zeigt eine vom Konfigurationsadvisor generierte Beispielausgabe.

Gehen Sie wie folgt vor, um den Konfigurationsadvisor auszuführen:

1. Stellen Sie eine Verbindung zur Datenbank PERSONL her, indem Sie den folgenden Befehl in die Befehlszeile eingeben:

   ```
   DB2 CONNECT TO PERSONL
   ```

2. Führen Sie den Befehl **AUTOCONFIGURE** über die Befehlszeile (CLP) mit Angaben zur Art und Weise der Verwendung der Datenbank aus. Wie im folgenden Beispiel gezeigt, können Sie die Option **APPLY** auf den Wert NONE setzen, um anzugeben, dass Sie die Konfigurationsempfehlungen nur prüfen, jedoch nicht anwenden möchten:

   ```
   DB2 AUTOCONFIGURE USING
        MEM_PERCENT 60
        WORKLOAD_TYPE MIXED
        NUM_STMTS 500
        ADMIN_PRIORITY BOTH
        IS_POPULATED YES
        NUM_LOCAL_APPS 0
        NUM_REMOTE_APPS 20
        ISOLATION RR
        BP_RESIZEABLE YES
   APPLY NONE
   ```

Wenn Sie sich über den Wert eines Parameters für den Befehl nicht im Klaren sind, können Sie ihn auslassen, sodass der Standardwert verwendet wird. Sie können bis zu 10 Parameter ohne Werte übergeben: MEM_PERCENT, WORKLOAD_TYPE usw., wie im vorigen Beispiel gezeigt.

Die durch den Befehl **AUTOCONFIGURE** generierten Empfehlungen werden auf dem Bildschirm im Tabellenformat wie in Abb. 2 dargestellt angezeigt:

```
       Frühere und angewendete Werte für die Datenbankmanagerkonfiguration
Beschreibung                                                 Parameter   Aktueller Wert  Empfohlener Wert
--------------------------------------------------------------------------------------------------
Zwischenspeicher für Anw.unterstützungsebene (4 KB)             (ASLHEAPSZ) = 15              15
Anzahl Kommunikationspuffer (4 KB)                        (FCM_NUM_BUFFERS) = AUTOMATIC       AUTOMATIC
Partitionsinterne Parallelität aktivieren                  (INTRA_PARALLEL) = NO              NO
Max. Grad der Parallelität bei Abfragen                   (MAX_QUERYDEGREE) = ANY             1
Größe des Agentenpools                                     (NUM_POOLAGENTS) = 100(berechnet)  200
Ursprüngliche Anzahl Agenten im Pool                       (NUM_INITAGENTS) = 0               0
Max. E/A-Blockgröße (Byte) für Requester                         (RQRIOBLK) = 32767           32767
Schwellenwert für Sortierspeicher (4 KB)                     (SHEAPTHRES) = 0               0

       Frühere und angewendete Werte für die Datenbankkonfiguration
Beschreibung                                                 Parameter   Aktueller Wert  Empfohlener Wert
--------------------------------------------------------------------------------------------------
Standardzwischenspeicher für Anwendungen (4 KB)                 (APPLHEAPSZ) = 256             256
Katalogcachegröße (4 KB)                                  (CATALOGCACHE_SZ) = (MAXAPPLS*4)    260
Schwellenwert für geänderte Seiten                         (CHNGPGS_THRESH) = 60              80
Zwischenspeicher der Datenbank (4 KB)                             (DBHEAP) = 1200            2791
Grad der Parallelität                                         (DFT_DEGREE) = 1               1
Standardwert für Speicherbereichsgröße (Seiten)             (DFT_EXTENT_SZ) = 32              32
Standardwert für Vorablesezugriffsgröße (Seiten)           (DFT_PREFETCH_SZ) = AUTOMATIC       AUTOMATIC
Standardabfrageoptimierungsklasse                             (DFT_QUERYOPT) = 5               5
Max. Speicher für Sperrenliste (4 KB)                             (LOCKLIST) = 100             AUTOMATIC
Protokollpuffergröße (4 KB)                                       (LOGBUFSZ) = 8               99
Protokolldateigröße (4 KB)                                       (LOGFILSIZ) = 1000            1024
Anzahl primärer Protokolldateien                                (LOGPRIMARY) = 3               8
Anzahl sekundärer Protokolldateien                               (LOGSECOND) = 2               3
Max. Anzahl aktiver Anwendungen                                   (MAXAPPLS) = AUTOMATIC       AUTOMATIC
Anzahl der Sperrenlisten pro Anwend. (in %)                       (MAXLOCKS) = 10              AUTOMATIC
Anzahl Gruppencommits                                            (MINCOMMIT) = 1               1
Anzahl asynchroner Seitenlöschfunktionen                   (NUM_IOCLEANERS) = 1               1
Anzahl von E/A-Servern                                      (NUM_IOSERVERS) = 3               4
Größe des Paketcache  (4 KB)                                   (PCKCACHESZ) = (MAXAPPLS*8)    1533
Vor bedingtem Prüfpunkt zu schreibende Protokollsätze (%)         (SOFTMAX) = 100             320
Zwischenspeicher für Sortierlisten (4 KB)                        (SORTHEAP) = 256             AUTOMATIC
Anweisungszwischenspeicher (4 KB)                                (STMTHEAP) = 4096            4096
Größe des Statistikzwischenspeichers (4 KB)                 (STAT_HEAP_SZ) = 4384            4384
Zwischenspeicher für Dienstprogramme (4 KB)                 (UTIL_HEAP_SZ) = 5000            113661
Speicher mit automatischer Leistungsoptimierung            (SELF_TUNING_MEM) = ON              ON
    Automatische Statistikerstellung                        (AUTO_RUNSTATS) = ON              ON
Sortierspeicherschwelle für gemeinsame Sortierungen (4 KB) (SHEAPTHRES_SHR) = 5000            AUTOMATIC

       Frühere und angewendete Werte für Pufferpool(s)
Beschreibung                                                 Parameter   Aktueller Wert  Empfohlener Wert
--------------------------------------------------------------------------------------------------
IBMDEFAULTBP                                              Pufferpoolgröße = -2              340985

DB210203I AUTOCONFIGURE wurde erfolgreich beendet. Die Werte für den Datenbankmanager
oder die Datenbankkonfiguration wurden möglicherweise geändert. Die Instanz muss neu
gestartet werden, damit die Änderungen wirksam werden. Außerdem sollten Sie für die
Pakete einen Rebind durchführen, sobald die neuen Konfigurationsparameter wirksam sind.
```

Abbildung 2. Beispielausgabe des Konfigurationsadvisors

Wenn Sie allen Empfehlungen zustimmen, können Sie den Befehl **AUTOCONFIGURE** entweder erneut ausführen und dabei über die Option APPLY angeben, dass die empfohlenen Werte angewendet werden sollen, oder einzelne Konfigurationsparameter mit dem Befehl **UPDATE DATABASE MANAGER CONFIGURATION** und dem Befehl **UPDATE DATABASE CONFIGURATION** aktualisieren.

Drosselung von Dienstprogrammen

Die Dienstprogrammdrosselung reguliert die Beeinträchtigung der Leistung durch Wartungsdienstprogramme, sodass sie gleichzeitig innerhalb von Produktionszeiträumen ausgeführt werden können. Obwohl die Richtlinie für die Auslastungswirkung (eine Einstellung, die die Ausführung von Dienstprogrammen in einem gedrosselten Modus ermöglicht) standardmäßig definiert wird, müssen Sie die Priorität der Auslastungswirkung festlegen. Dies ist eine Einstellung jeder Bereinigungsfunktion, die die Drosselungspriorität angibt, wenn Sie ein Dienstprogramm ausführen, das gedrosselt werden soll.

Das Drosselungssystem stellt sicher, dass die gedrosselten Dienstprogramme so häufig wie möglich ausgeführt werden, ohne gegen die Richtlinie für die Auslastungswirkung zu verstoßen. Sie können Operationen zur Statistikerfassung, Backup-Operationen, Datenumverteilungen und asynchrone Indexbereinigungen drosseln.

Sie definieren die Richtlinie für die Auslastungswirkung mithilfe des Konfigurationsparameters **util_impact_lim**.

Bereinigungsfunktionen sind in die Drosselungseinrichtung für Dienstprogramme integriert. Standardmäßig ist jede Bereinigungsfunktion (für Indizes) auf den Wert 50 für die Priorität der Auslastungswirkung von Dienstprogrammen eingestellt. (Akzeptable Werte liegen zwischen 1 und 100, wobei 0 keine Drosselung angibt.) Sie können die Priorität mit dem Befehl **SET UTIL_IMPACT_PRIORITY** oder mit der API 'db2UtilityControl' ändern.

Asynchrone Indexbereinigung

Als asynchrone Indexbereinigung (AIC, Asynchronous Index Cleanup) wird die verzögerte Bereinigung von Indizes nach Operationen bezeichnet, die Indexeinträge ungültig machen. Abhängig vom Typ des Index können die Einträge Satz-IDs (RIDs) oder Block-IDs (BIDs) sein. Ungültige Indexeinträge werden von Indexbereinigungsfunktionen entfernt, die asynchron im Hintergrund ausgeführt werden.

Die asynchrone Indexbereinigung beschleunigt den Prozess des Aufhebens der Zuordnung einer Datenpartition zu einer partitionierten Tabelle und wird eingeleitet, wenn die partitionierte Tabelle mindestens einen nicht partitionierten Index hat. In diesem Fall entfernt die asynchrone Indexbereinigung alle nicht partitionierten Indexeinträge, die sich auf die Datenpartition mit aufgehobener Zuordnung beziehen, und außerdem alle pseudo-gelöschten Einträge. Wenn alle Indizes bereinigt sind, wird die Kennung (ID), die zu der Datenpartition mit aufgehobener Zuordnung gehört, aus dem Systemkatalog entfernt. In DB2 Version 9.7 Fixpack 1 und späteren Releases wird die asynchrone Indexbereinigung durch eine Task zur asynchronen Aufhebung von Partitionszuordnungen eingeleitet.

Wenn in einer Version vor DB2 Version 9.7 Fixpack 1 die partitionierte Tabelle abhängige MQTs (Materialized Query Tables) hat, wird die asynchrone Indexbereinigung erst nach der Ausführung einer Anweisung SET INTEGRITY ausgeführt.

Der normale Tabellenzugriff bleibt während der Ausführung der asynchronen Indexbereinigung erhalten. Abfragen, die auf die Indizes zugreifen, ignorieren alle ungültigen Einträge, die noch nicht bereinigt wurden.

In den meisten Fällen wird eine Bereinigungsfunktion für jeden nicht partitionierten Index gestartet, der der partitionierten Tabelle zugeordnet ist. Ein interner Taskverteilungsdämonprozess ist für die Verteilung der AIC-Tasks an die jeweiligen Tabellenpartitionen sowie für die Zuweisung von Datenbankagenten zuständig. Der Verteilungsdämonprozess und die Bereinigungsagenten sind interne Systemanwendungen, die in der Ausgabe des Befehls **LIST APPLICATIONS** mit den Anwendungsnamen **db2taskd** bzw. **db2aic** angegeben werden. Zur Vermeidung versehentlicher Unterbrechungen lässt sich der Abbruch von Systemanwendungen nicht erzwingen. Der Verteilungsdämon bleibt online, solange die Datenbank aktiv ist. Die Bereinigungsfunktionen bleiben aktiv, bis die Bereinigung abgeschlossen ist. Falls die Datenbank während der Bereinigung inaktiviert wird, nimmt die AIC die Arbeit wieder auf, wenn Sie die Datenbank reaktivieren.

Auswirkung der asynchronen Indexbereinigung auf die Leistung

Die asynchrone Indexbereinigung wirkt sich nur minimal auf die Leistung aus.

Ein sofortiger Zeilensperrentest ist erforderlich, um festzustellen, ob ein pseudogelöschter Eintrag festgeschrieben wurde. Da die Sperre jedoch nie aktiviert wird, wird der gemeinsame Zugriff nicht beeinträchtigt.

Jede Bereinigungsfunktion aktiviert eine minimale Tabellenbereichssperre (IX) und eine Tabellensperre (IS). Diese Sperren werden freigegeben, wenn eine Bereinigungsfunktion feststellt, dass andere Anwendungen auf Sperren warten. Wenn dieser Fall eintritt, setzt die Bereinigungsfunktion die Verarbeitung für fünf Minuten aus.

Bereinigungsfunktionen sind in die Drosselungseinrichtung für Dienstprogramme integriert. Standardmäßig ist für jede Bereinigungsfunktion der Wert 50 für die Priorität der Auslastungswirkung von Dienstprogrammen eingestellt. Sie können die Priorität mit dem Befehl **SET UTIL_IMPACT_PRIORITY** oder mit der API db2UtilityControl ändern.

Überwachen der asynchronen Indexbereinigung

Sie können die asynchrone Indexbereinigung mit dem Befehl **LIST UTILITIES** überwachen. Jede Indexbereinigungsfunktion wird in der Ausgabe als separates Dienstprogramm aufgeführt. Das folgende Beispiel zeigt eine Ausgabe des Befehls **LIST UTILITIES SHOW DETAIL**:

```
ID                                   = 2
Typ                                  = ASYNCHRONOUS INDEX CLEANUP
Datenbankname                        = WSDB
Partitionsnummer                     = 0
Beschreibung                         = Tabelle: USER1.SALES, Index: USER1.I2
Startzeit                            = 2005-12-15 11:15:01.967939
Status                               = Wird ausgeführt
Aufruftyp                            = Automatisch
Drosselung:
   Priorität                         = 50
Fortschrittsüberwachung:
      Gesamte Arbeit                 = 5 Seiten
      Abgeschlossene Arbeit          = 0 Seiten
      Startzeit                      = 2005-12-15 11:15:01.979033
```

```
ID                              = 1
Typ                             = ASYNCHRONOUS INDEX CLEANUP
Datenbankname                   = WSDB
Partitionsnummer                = 0
Beschreibung                    = Tabelle: USER1.SALES, Index: USER1.I1
Startzeit                       = 2005-12-15 11:15:01.978554
Status                          = Wird ausgeführt
Aufruftyp                       = Automatisch
Drosselung:
   Priorität                    = 50
Fortschrittsüberwachung:
      Gesamte Arbeit            = 5 Seiten
      Abgeschlossene Arbeit     = 0 Seiten
      Startzeit                 = 2005-12-15 11:15:01.980524
```

In gezeigten Fall sind zwei Bereinigungsfunktionen an der Tabelle USERS1.SALES aktiv. Die eine Bereinigungsfunktion bearbeitet den Index I1, die andere den Index I2. Dem Abschnitt unter 'Fortschrittsüberwachung' ist die geschätzte Gesamtzahl von zu bereinigenden Indexseiten sowie die aktuelle Anzahl der bereits bereinigten Indexseiten zu entnehmen.

Das Feld `Status` gibt den aktuellen Status einer Bereinigungsfunktion an. Der normale Status ist 'Wird ausgeführt'. Jedoch kann sich die Bereinigungsfunktion 'Im Wartestatus' befinden, wenn sie darauf wartet, einem verfügbaren Datenbankagenten zugeordnet zu werden oder wenn sie wegen eines Sperrenkonflikts vorübergehend ausgesetzt ist.

Beachten Sie, dass verschiedene Tasks in verschiedenen Datenbankpartitionen die gleiche Dienstprogramm-ID haben können, da jede Datenbankpartition IDs den Tasks zuordnet, die in dieser Datenbankpartition ausgeführt werden.

Asynchrone Indexbereinigung für MDC-Tabellen

Sie können die Leistung einer Rolloutlöschung durch die Nutzung einer asynchronen Indexbereinigung (AIC, Asynchronous Index Cleanup) verbessern. Eine Rolloutlöschung ist eine effiziente Methode zum Löschen von Datenblöcken, die bestimmten Kriterien entsprechen, aus Tabellen mit mehrdimensionalem Clustering (MDC). Die asynchrone Indexbereinigung ist die verzögerte Bereinigung von Indizes, die im Anschluss an Operationen erfolgt, durch die Indexeinträge ungültig werden.

Indizes werden synchron bei einer Standardrolloutlöschung bereinigt. Wenn eine Tabelle viele Satz-ID-Indizes (RID-Indizes) hat, fällt ein beträchtlicher Zeitaufwand für das Entfernen von Indexschlüsseln an, die auf die Tabellenzeilen verweisen, die gelöscht werden. Sie können den Rollout beschleunigen, indem Sie angeben, dass diese Indizes nach dem Festschreiben der Löschoperation zu bereinigen sind.

Zur Nutzung der asynchronen Indexbereinigung für MDC-Tabellen müssen Sie den Mechanismus für den *Rollout mit verzögerter Indexbereinigung* explizit aktivieren. Ein Rollout mit verzögerter Indexbereinigung kann durch zwei Methoden angegeben werden: Durch Setzen der Registrierdatenbankvariablen **DB2_MDC_ROLLOUT** auf den Wert `DEFER` und durch Ausführen der Anweisung SET CURRENT MDC ROLLOUT MODE. Während einer Rolloutoperation mit verzögerter Indexbereinigung werden Blöcke als durch Rollout gelöscht markiert, wobei die Aktualisierung an den Satz-ID-Indizes erst erfolgt, wenn die Transaktion festgeschrieben wurde. Block-ID-Indizes (BID-Indizes) werden während der Löschoperation bereinigt, weil sie keine Verarbeitung auf Zeilenebene erfordern.

Die Rolloutoperation der verzögerten Indexbereinigung wird aufgerufen, wenn eine Rolloutlöschung festgeschrieben wird oder, falls die Datenbank beendet wurde, wenn auf die Tabelle nach dem Datenbankneustart zum ersten Mal zugegriffen wird. Während der Ausführung der asynchronen Indexbereinigung werden Abfragen, die auf die Indizes, einschließlich des Index, der gerade bereinigt wird, zugreifen, erfolgreich ausgeführt.

Pro MDC-Tabelle ist eine koordinierende Bereinigungsfunktion vorhanden. Die Indexbereinigung für mehrere Rollouts wird innerhalb der Bereinigungsfunktion konsolidiert, die wiederum einen Bereinigungsagenten für jeden Satz-ID-Index startet. Bereinigungsagenten aktualisieren die Satz-ID-Indizes parallel. Bereinigungsfunktionen sind darüber hinaus in die Drosselungseinrichtung für Dienstprogramme integriert. Standardmäßig ist jede Bereinigungsfunktion auf den Wert 50 für die Priorität der Auslastungswirkung von Dienstprogrammen eingestellt. (Akzeptable Werte liegen zwischen 1 und 100, wobei 0 keine Drosselung angibt.) Sie können diese Priorität mit dem Befehl **SET UTIL_IMPACT_PRIORITY** oder mit der API db2UtilityControl ändern.

Anmerkung: In DB2 Version 9.7 und späteren Releases wird ein Rollout mit verzögerter Bereinigung für eine datenpartitionierte MDC-Tabelle mit partitionierten Satz-ID-Indizes nicht unterstützt. Es werden nur die Modi NONE und IMMEDIATE unterstützt. Der Modus des Rollouts mit Bereinigung ist IMMEDIATE, wenn die Registrierdatenbankvariable **DB2_MDC_ROLLOUT** auf den Wert DEFER gesetzt ist oder wenn das Sonderregister CURRENT MDC ROLLOUT MODE auf den Wert DEFERRED gesetzt ist, um die Einstellung der Variablen **DB2_MDC_ROLLOUT** zu überschreiben.

Wenn nur nicht partitionierte Satz-ID-Indizes für die MDC-Tabelle vorhanden sind, wird ein Rollout mit verzögerter Indexbereinigung unterstützt. MDC-Blockindizes können partitioniert oder nicht partitioniert sein.

Überwachen des Fortschritts einer Rolloutoperation mit verzögerter Indexbereinigung

Da die durch Rollout gelöschten Blöcke in einer MDC-Tabelle erst wieder verwendet werden können, wenn die Bereinigung abgeschlossen ist, ist es nützlich, den Fortschritt einer Rolloutoperation mit verzögerter Indexbereinigung zu überwachen. Mit dem Befehl **LIST UTILITIES** können Sie einen Dienstprogrammüberwachungseintrag für jeden Index anzeigen, der momentan bereinigt wird. Sie können auch mit der Tabellenfunktion ADMIN_GET_TAB_INFO oder mit dem Befehl **GET SNAPSHOT** die Gesamtzahl der MDC-Tabellenblöcke in der Datenbank abrufen, für die eine asynchrone Bereinigung nach einer Rolloutlöschung ansteht (BLOCKS_PENDING_CLEANUP).

In der folgenden Beispielausgabe für den Befehl **LIST UTILITIES SHOW DETAIL** wird der Fortschritt durch die Anzahl der Seiten in jedem Index angezeigt, die bereinigt wurden. Jede Phase stellt einen Satz-ID-Index dar.

```
ID                                = 2
Typ                               = MDC ROLLOUT INDEX CLEANUP
Datenbankname                     = WSDB
Partitionsnummer                  = 0
Beschreibung                      = TABLE.<schemaname>.<tabellenname>
Startzeit                         = 06-12-2006 08:56:33.390158
Status                            = Wird ausgeführt
Aufruftyp                         = Automatisch
Drosselung:
   Priorität                      = 50
```

```
Fortschrittsüberwachung:
   Geschätzte Fertigstellung (%) = 83
   Phasennummer                  = 1
      Beschreibung               = <schemaname>.<indexname>
      Gesamte Arbeit             = 13 Seiten
      Abgeschlossene Arbeit      = 13 Seiten
      Startzeit                  = 06-12-2006 08:56:33.391566
   Phasennummer                  = 2
      Beschreibung               = <schemaname>.<indexname>
      Gesamte Arbeit             = 13 Seiten
      Abgeschlossene Arbeit      = 13 Seiten
      Startzeit                  = 06-12-2006 08:56:33.391577
   Phasennummer                  = 3
      Beschreibung               = <schemaname>.<indexname>
      Gesamte Arbeit             = 9 Seiten
      Abgeschlossene Arbeit      = 3 Seiten
      Startzeit                  = 06-12-2006 08:56:33.391587
```

Kapitel 4. Instanzen

Eine *Instanz* ist eine logische Datenbankmanagerumgebung, in der Sie Datenbanken katalogisieren und Konfigurationsparameter definieren. Bei Bedarf können Sie mehrere Instanzen auf demselben physischen Server erstellen und für jede Instanz eine spezifische Datenbankserverumgebung bereitstellen.

Anmerkung: Für Nichtrootinstallationen unter Linux und UNIX-Betriebssystemen wird nur eine Instanz bei der Installation Ihres DB2-Produkts erstellt. Es können keine weiteren Instanzen erstellt werden.

Mehrere Instanzen können beispielsweise zu folgenden Zwecken wünschenswert sein:

- Betreiben einer Instanz für die Entwicklungsumgebung und einer weiteren für die Geschäftsumgebung.
- Anpassen einer Instanz für eine bestimmte Umgebung.
- Begrenzen des Zugriffs auf sensible Daten.
- Steuern der Erteilung der Berechtigung SYSADM, SYSCTRL und SYSMAINT für jede Instanz.
- Optimieren der Datenbankmanagerkonfiguration für jede Instanz.
- Begrenzen der durch den Ausfall einer Instanz entstehenden Auswirkungen. Im Fall eines Ausfalls ist jeweils nur eine Instanz betroffen. Die übrigen Instanzen funktionieren weiterhin ordnungsgemäß.

Mehrere Instanzen stellen zusätzliche Anforderungen:

- Für jede Instanz sind zusätzliche Systemressourcen (virtueller Speicher und Plattenspeicherplatz) erforderlich.
- Der Verwaltungsaufwand ist größer, da mehrere Instanzen zu verwalten sind.

Im Instanzverzeichnis werden alle Informationen für eine Datenbankinstanz gespeichert. Sobald das Instanzverzeichnis erstellt ist, können Sie dessen Position nicht mehr ändern. Das Verzeichnis enthält folgende Objekte:

- Die Konfigurationsdatei des Datenbankmanagers
- Das Systemdatenbankverzeichnis
- Das Knotenverzeichnis
- Die Knotenkonfigurationsdatei (`db2nodes.cfg`)
- Alle weiteren Dateien mit Debuginformationen, wie z. B. den Speicherauszug der Ausnahme- und Registrierdaten oder den Aufrufstapel für die DB2-Datenbankprozesse.

Terminologie:

Bitlänge
Die Anzahl an Bits für den virtuellen Speicher: 32-Bit und 64-Bit sind am meisten verbreitet. Dieser Begriff wird möglicherweise verwendet, um auf die Bitlänge einer Instanz, eines Anwendungscodes oder eines Codes einer externen Routine zu verweisen. 32-Bit-Anwendung bedeutet dasselbe wie Anwendung mit einer Länge von 32-Bit.

32-Bit-DB2-Instanz
Eine DB2-Instanz, die alle 32-Bit-Binärdaten enthält, einschließlich gemeinsam genutzter 32-Bit-Bibliotheken und ausführbarer 32-Bit-Dateien.

64-Bit-DB2-Instanz
Eine DB2-Instanz, die gemeinsam genutzte 64-Bit-Bibliotheken und gemeinsam genutzte 64-Bit-Dateien sowie alle 32-Bit-Clientanwendungsbibliotheken (für Client und Server) und Unterstützung für externe 32-Bit-Routinen (nur für eine Serverinstanz) enthält.

Entwerfen von Instanzen

DB2-Datenbanken werden innerhalb von DB2-Instanzen auf dem Datenbankserver erstellt. Durch die Erstellung mehrerer Instanzen auf demselben physischen Server wird für jede Instanz eine exklusive Datenbankserverumgebung bereitgestellt.

Sie können zum Beispiel eine Testumgebung und eine Produktionsumgebung auf demselben Computer verwalten. Sie können auch eine Instanz für jede Anwendung erstellen und sie ausschließlich für die Anwendung, der sie dienen soll, optimieren. Oder Sie können Ihre Lohndatenbank zwecks Datenschutz in einer eigenen Instanz speichern, sodass Eigner anderer Instanzen (auf demselben Server) die Lohndaten nicht anzeigen können.

Der Installationsprozess erstellt eine DB2-Standardinstanz, die durch die Umgebungsvariable DB2INSTANCE definiert wird. Diese Instanz wird für die meisten Operationen verwendet. Jedoch können auch nach der Installation Instanzen erstellt (bzw. gelöscht) werden.

Beim Festlegen und Entwerfen der Instanzen für Ihre Umgebung müssen Sie beachten, dass jede Instanz den Zugriff auf eine oder mehrere Datenbanken steuert. Jede Datenbank in einer Instanz hat einen eindeutigen Namen, besitzt eine eigene Gruppe von Systemkatalogtabellen (die zur Verwaltung der in der Datenbank erstellten Objekte dienen) und verfügt über eine eigene Konfigurationsdatei. Darüber hinaus hat jede Datenbank eine eigene Gruppe von erteilbaren Berechtigungen und Zugriffsrechten, die steuern, wie Benutzer mit den in der Datenbank gespeicherten Daten und Datenbankobjekten interagieren. Abb. 3 auf Seite 71 zeigt die hierarchische Beziehung zwischen Systemen, Instanzen und Datenbanken.

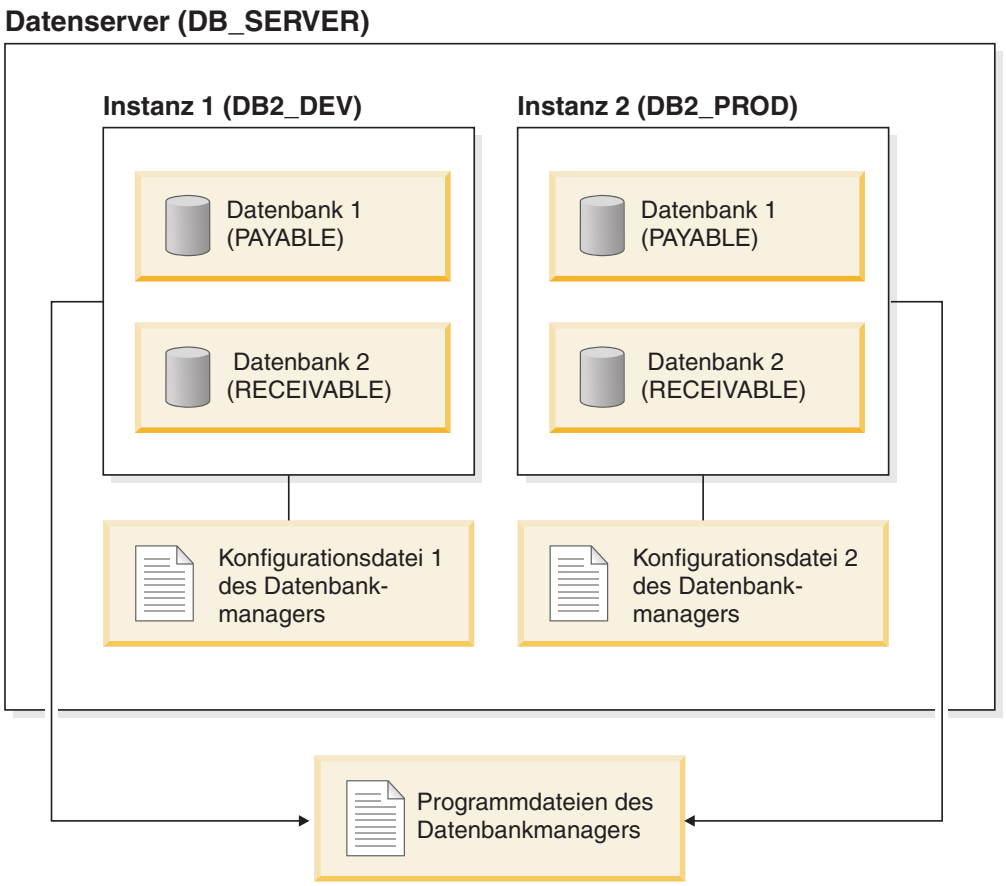


Abbildung 3. Hierarchische Beziehung zwischen DB2-Systemen, -Instanzen und -Datenbanken

Darüber hinaus müssen Sie auch einen weiteren besonderen Typ von Instanz in Ihre Überlegungen mit einbeziehen, der als *DB2-Verwaltungsserver* (DAS - DB2 Administration Server) bezeichnet wird. Der DAS ist ein spezieller DB2-Verwaltungssteuerpunkt, der nur zur Unterstützung der Verwaltungstasks auf anderen DB2-Servern verwendet wird. Ein DAS muss ausgeführt werden, wenn Sie 'Clientkonfiguration - Unterstützung' zum Erkennen der fernen Datenbanken verwenden möchten oder die grafischen Tools, die mit dem DB2-Produkt geliefert werden (z. B. IBM Data Studio). In einem DB2-Datenbankserver ist auch bei mehreren Instanzen immer nur ein DAS vorhanden.

Wichtig: Der DB2-Verwaltungsserver (DAS) gilt in Version 9.7 als veraltet und wird möglicherweise in einem zukünftigen Release entfernt. Der DAS wird in DB2 pureScale-Umgebungen nicht unterstützt. Verwenden Sie Softwareprogramme, die das Secure Shell-Protokoll für die Fernverwaltung nutzen. Weitere Informationen hierzu finden Sie im Abschnitt „ DB2-Verwaltungsserver (DAS) gilt als veraltet” in http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0059276.html.

Wenn Ihre Instanzen erstellt sind, können Sie zu jeder anderen verfügbaren Instanz (einschließlich der Instanzen auf anderen Systemen) eine Verbindung mit dem Befehl ATTACH herstellen. Wenn diese Verbindung besteht, können Sie Wartungs- und Dienstprogrammoperationen ausführen, die nur auf der Instanzebene ausgeführt werden können. Zum Beispiel können Sie eine Datenbank erstellen, Anwen-

dung zwangsweise von einer Datenbank trennen, Datenbankaktivitäten überwachen oder den Inhalt der Konfigurationsdatei des Datenbankmanagers ändern, die einer bestimmten Instanz zugeordnet ist.

Standardinstanz

Im Rahmen Ihrer DB2-Installationsprozedur erstellen Sie eine Erstinstanz des Datenbankmanagers, die den Namen DB2 besitzt, sofern nicht bereits eine andere Instanz mit dem Namen „DB2" vorhanden ist. Wenn DB2 Version 8 installiert ist und Sie ein Upgrade auf Version 9.1 oder Version 9.5 durchführen, hat die Standardinstanz den Namen „DB2_01".

Unter Linux und UNIX kann die Erstinstanz einen beliebigen, den geltenden Namensregeln entsprechenden Namen erhalten. Der Instanzname wird zum Definieren der Verzeichnisstruktur verwendet.

Damit diese Instanz sofort verwendet werden kann, werden bei der Installation folgende Konfigurationswerte festgelegt:

- Die Umgebungsvariable **DB2INSTANCE** wird auf den Wert DB2 gesetzt.
- Die Registrierdatenbankvariable **DB2INSTDEF** wird auf den Wert DB2 gesetzt.

Diese Einstellungen legen „DB2" als Standardinstanz fest. Sie können die Standardinstanz erst ändern, nachdem Sie eine weitere Instanz erstellt haben.

Vor der Verwendung des Datenbankmanagers muss die Datenbankumgebung für jeden Benutzer aktualisiert werden, sodass sie auf eine Instanz zugreifen und die DB2-Datenbankprogramme ausführen kann. Dies gilt für alle Benutzer (einschließlich der Benutzer mit Verwaltungsaufgaben).

Unter Linux- und UNIX-Betriebssystemen werden Beispielscriptdateien bereitgestellt, die Sie beim Einrichten der Datenbankumgebung unterstützten. Folgende Dateien werden bereitgestellt: `db2profile` für die Bourne- oder Korn-Shell und `db2cshrc` für die C-Shell. Diese Scripts befinden sich im Unterverzeichnis `sqllib` im Ausgangsverzeichnis des Instanzeigners. Der Instanzeigner oder jeder beliebige Benutzer, der zur SYSADM-Gruppe der Instanz gehört, kann das Script für alle Benutzer einer Instanz anpassen. Verwenden Sie `sqllib/userprofile` und `sqllib/usercshrc`, um ein Script für die einzelnen Benutzer anzupassen.

In den leeren Dateien `sqllib/userprofile` und `sqllib/usercshrc` können Sie bei der Instanzerstellung eigene Einstellungen für die Instanzumgebung hinzufügen. Die Dateien `db2profile` und `db2cshrc` werden bei der Instanzaktualisierung einer DB2-Fixpackinstallation überschrieben. Wenn Sie die neuen Umgebungseinstellungen in den Scripts `db2profile` oder `db2cshrc` nicht verwenden möchten, können Sie sie mit dem entsprechenden Script 'user' überschreiben, das am Ende des Scripts `db2profile` bzw. `db2cshrc` aufgerufen wird. Während des Upgrades einer Instanz (mit dem Befehl **db2iupgrade**) werden die Scripts 'user' mit kopiert, sodass Ihre Umgebungsänderungen weiterhin verwendet werden.

Das Beispielscript enthält Anweisungen für folgende Aufgaben:

- Aktualisieren eines Benutzerpfads (**PATH**) durch Hinzufügen der folgenden Verzeichnisse zum vorhandenen Suchpfad: die Unterverzeichnisse `bin`, `adm` und `misc` im Unterverzeichnis `sqllib` im Ausgangsverzeichnis des Instanzeigners
- Setzen der Umgebungsvariablen **DB2INSTANCE** auf den Instanznamen

Instanzverzeichnis

Im Instanzverzeichnis werden alle Informationen für eine Datenbankinstanz gespeichert. Die Position des Instanzverzeichnisses lässt sich nach der Erstellung nicht mehr ändern.

Das Instanzverzeichnis hat folgenden Inhalt:

- Die Konfigurationsdatei des Datenbankmanagers
- Das Systemdatenbankverzeichnis
- Das Knotenverzeichnis
- Die Knotenkonfigurationsdatei (`db2nodes.cfg`)
- Weitere Dateien mit Debuginformationen, wie z. B. den Speicherauszug der Ausnahme- und Registrierdaten oder den Aufrufstapel für die DB2-Prozesse.

Unter Linux- und UNIX-Betriebssystemen befindet sich das Instanzverzeichnis im Verzeichnis *INSTHOME*`/sqllib`, wobei *INSTHOME* das Ausgangsverzeichnis des Instanzeigners ist. Der Standardinstanz kann ein beliebiger Name nach den Richtlinien der Namensregeln gegeben werden.

Unter Windows-Betriebssystemen befindet sich das Instanzverzeichnis unter dem Verzeichnis `/sqllib`, in dem das DB2-Datenbankprodukt installiert wurde. Der Instanzname ist mit dem Namen des Service identisch. Daher sollte dieser nicht mit anderen Namen in Konflikt stehen. Kein Instanzname darf mit einem anderen Servicenamen identisch sein. Sie müssen über die richtige Berechtigung verfügen, um einen Service zu erstellen.

In einer Umgebung mit partitionierten Datenbanken wird das Instanzverzeichnis von allen Datenbankpartitionsservern, die zu der Instanz gehören, gemeinsam genutzt. Darum muss das Instanzverzeichnis auf einem gemeinsam genutzten Netzlaufwerk erstellt werden, auf das alle Computer in der Instanz zugreifen können.

db2nodes.cfg

Die Datei `db2nodes.cfg` dient zur Definition der Datenbankpartitionsserver, die an einer DB2-Instanz beteiligt sind. Darüber hinaus dient die Datei `db2nodes.cfg` zur Angabe der IP-Adresse oder des Hostnamens einer Hochgeschwindigkeitsverbindung, wenn Sie eine Hochgeschwindigkeitsverbindung zur Kommunikation für die Datenbankpartitionsserver verwenden möchten.

Mehrere Instanzen (Linux, UNIX)

Es ist möglich, unter einem Linux oder UNIX-Betriebssystem mehrere Instanzen zu haben, wenn das DB2-Produkt mit Rootberechtigungen installiert wurde. Obwohl alle Instanzen gleichzeitig ausgeführt werden, arbeiten diese unabhängig voneinander. Sie können jedoch immer nur jeweils innerhalb einer einzigen Instanz des Datenbankmanagers arbeiten.

Anmerkung: Zur Vermeidung von Umgebungskonflikten zwischen zwei oder mehr Instanzen sollten Sie sicherstellen, dass jede Instanz über ein eigenes Ausgangsverzeichnis verfügt. Eine gemeinsame Nutzung des Ausgangsverzeichnisses führt zur Rückgabe von Fehlern. Jedes Ausgangsverzeichnis kann sich im selben oder in einem anderen Dateisystem befinden.

Jeder Instanz wird der Instanzeigner und die Systemverwaltungsgruppe (SYSADM) zugeordnet. Dies geschieht bei der Erstellung der Instanz. Eine Benutzer-ID

bzw. ein Benutzername kann nur für jeweils eine Instanz verwendet werden. Diese Benutzer-ID bzw. dieser Benutzername wird auch als der *Instanzeigner* bezeichnet.

Jeder Instanzeigner muss über ein eindeutiges Ausgangsverzeichnis verfügen. Alle zum Ausführen der Instanz erforderlichen Konfigurationsdateien werden im Ausgangsverzeichnis der Benutzer-ID bzw. des Benutzernamens des Instanzeigners erstellt. Falls es erforderlich wird, die Benutzer-ID bzw. den Benutzernamen des Instanzeigners aus dem System zu entfernen, könnten Sie der Instanz zugeordnete Dateien sowie den Zugriff auf in dieser Instanz gespeicherte Daten verlieren. Daher sollten Sie die Benutzer-ID bzw. den Benutzernamen eines Instanzeigners ausschließlich für die Ausführung des Datenbankmanagers dedizieren.

Die Primärgruppe des Instanzeigners ist ebenfalls von Bedeutung. Diese Primärgruppe wird automatisch zur Systemverwaltungsgruppe der Instanz und erhält die Berechtigung SYSADM für die Instanz. Andere Benutzer-IDs oder Benutzernamen, die Mitglieder der Primärgruppe der Instanz sind, erhalten ebenfalls diese Berechtigungsstufe. Aus diesem Grund kann es wünschenswert sein, die Benutzer-ID bzw. den Benutzernamen des Instanzeigners einer Primärgruppe zuzuordnen, die für die Verwaltung von Instanzen reserviert ist. (Stellen Sie außerdem sicher, dass der Benutzer-ID bzw. dem Benutzernamen des Instanzeigners eine Primärgruppe zugeordnet ist. Andernfalls wird die Standardprimärgruppe des Systems verwendet.)

Wenn Sie bereits über eine Gruppe verfügen, die Sie als Systemverwaltungsgruppe für die Instanz verwenden wollen, können Sie diese Gruppe beim Erstellen der Benutzer-ID bzw. des Benutzernamens des Instanzeigners als Primärgruppe zuordnen. Fügen Sie andere Benutzer, denen Sie die Verwaltungsberechtigung für die Instanz erteilen wollen, derjenigen Gruppe hinzu, die als Systemverwaltungsgruppe zugeordnet ist.

Wenn Sie die Berechtigung SYSADM für die einzelnen Instanzen voneinander trennen wollen, stellen Sie sicher, dass jede Benutzer-ID bzw. jeder Benutzername eines Instanzeigners eine andere Primärgruppe verwendet. Bei Verwendung einer allgemeinen Berechtigung SYSADM für mehrere Instanzen können Sie jedoch dieselbe Primärgruppe für mehrere Instanzen verwenden.

Mehrere Instanzen (Windows)

Es ist möglich, mehrere Instanzen des DB2-Datenbankmanagers auf demselben Computer auszuführen. Jede Instanz des Datenbankmanagers pflegt eigene Datenbanken und verfügt über eigene Konfigurationsparameter für den Datenbankmanager.

Anmerkung: Die Instanzen können auch unterschiedlichen DB2-Kopien auf einem Computer angehören, der unterschiedliche Versionen des Datenbankmanagers haben kann. Wenn Sie ein 64-Bit-Windows-System verwenden, können Sie eine 32-Bit-DB2-Version oder eine 64-Bit-DB2-Version installieren, diese können jedoch nicht auf derselben Maschine koexistieren.

Eine Instanz des Datenbankmanagers besteht aus folgenden Komponenten:

- Ein Windows-Dienst (Service), der die Instanz darstellt. Der Name des Dienstes ist mit dem Namen der Instanz identisch. Der Anzeigename des Dienstes (im Fenster **Dienste**) ist der Instanzname, dem die Zeichenfolge „DB2 - ” vorangestellt ist. Zum Beispiel gibt es für eine Instanz mit dem Namen „DB2” einen Windows-Dienst mit dem Namen „DB2” der als „DB2 - *name_der_db2-kopie* - DB2” angezeigt wird.

Anmerkung: Für Clientinstanzen wird kein Windows-Dienst erstellt.

- Ein Instanzverzeichnis. Dieses Verzeichnis enthält die Konfigurationsdateien des Datenbankmanagers, das Systemdatenbankverzeichnis, das Knotenverzeichnis, das DCS-Verzeichnis, alle Diagnoseprotokolldateien sowie alle Speicherauszugsdateien, die der Instanz zugeordnet sind. Das Instanzverzeichnis ist je nach Edition des Betriebssystems der Windows-Familie unterschiedlich. Zur Feststellung des Standardverzeichnisses unter Windows prüfen Sie die Einstellung der Umgebungsvariablen **DB2INSTPROF** mithilfe des Befehls **db2set DB2INSTPROF**. Sie können das Standardinstanzverzeichnis auch ändern, indem Sie den Wert der Umgebungsvariablen **DB2INSTPROF** ändern. Wenn Sie es zum Beispiel auf `c:\DB2PROFS` setzen wollen, gehen Sie wie folgt vor:
  - Setzen Sie **DB2INSTPROF** mithilfe des Befehls **db2set.exe -g** auf den Wert `c:\DB2PROFS`.
  - Führen Sie den Befehl **DB2ICRT.exe** aus, um die Instanz zu erstellen.
- Wenn Sie unter Windows-Betriebssystemen eine Instanz erstellen, sind die Standardpositionen für Datendateien des Benutzers - wie zum Beispiel Instanzverzeichnisse und die Datei `db2cli.ini` - die folgenden Verzeichnisse:
  - Unter Windows XP und Windows 2003: `Dokumente und Einstellungen\All Users\Application Data\IBM\DB2\`*`Name_der_Kopie`*
  - Unter den Betriebssystemen Windows 2008 und Windows Vista (und späteren Versionen): `Program Data\IBM\DB2\`*`Name_der_Kopie`*

  Dabei ist *Name_der_Kopie* der Name der DB2-Kopie.

  Anmerkung: Die Position der Datei `db2cli.ini` kann sich ändern, je nachdem, ob der Microsoft ODBC Driver Manager verwendet wird, welcher Typ von Datenquellennamen (DSN) verwendet wird, welcher Typ von Client oder Treiber installiert ist und ob die Registrierdatenbankvariable **DB2CLIINIPATH** definiert ist.

Erstellen von Instanzen

Im Rahmen der Installation des Datenbankmanagers wird zwar eine Instanz erstellt, jedoch können Ihre Geschäftsanforderungen möglicherweise eine Erstellung zusätzlicher Instanzen erforderlich machen.

Vorbereitende Schritte

Wenn Sie der Administratorengruppe unter Windows angehören oder über Rootberechtigung auf Linux- oder UNIX-Betriebssystemen verfügen, können Sie zusätzliche Instanzen hinzufügen. Der Computer, auf dem die Instanz hinzugefügt wird, wird als Instanzeignercomputer (Knoten 0) definiert. Fügen Sie Instanzen unbedingt auf einem Computer hinzu, auf dem sich ein DB2-Verwaltungsserver (DAS) befindet. Die Instanz-IDs dürfen nicht 'root' sein und kein abgelaufenes Kennwort haben.

Einschränkungen

- Unter Linux- und UNIX-Betriebssystemen können zusätzliche Instanzen nicht für Nichtrootinstallationen erstellt werden.
- Wenn vorhandene Benutzer-IDs zur Erstellung von DB2-Instanzen verwendet werden, stellen Sie sicher, dass für die Benutzer-IDs folgende Bedingungen zutreffen:
  - Sie sind nicht gesperrt.
  - Sie haben keine abgelaufenen Kennwörter.

Vorgehensweise

Gehen Sie wie folgt vor, um eine Instanz über die Befehlszeile hinzuzufügen:

Geben Sie den Befehl db2icrt *instanzname* ein.
Bei der Erstellung einer Instanz auf einem AIX-Server müssen Sie die abgeschirmte Benutzer-ID (FencedID) angeben. Beispiel:

```
DB2DIR/instance/db2icrt -u db2fenc1 db2inst1
```

Bei der Verwendung des Befehls **db2icrt** zum Hinzufügen einer weiteren DB2-Instanz sollten Sie den Anmeldenamen des Instanzeigners und optional den Authentifizierungstyp der Instanz angeben. Der Authentifizierungstyp gilt für alle Datenbanken, die unter dieser Instanz erstellt werden. Der Authentifizierungstyp ist eine Anweisung, wo die Authentifizierung von Benutzern stattfinden soll.
Mit der Umgebungsvariablen **DB2INSTPROF** können Sie die Position des in **DB2PATH** angegebenen Instanzverzeichnisses ändern. Sie benötigen dafür Schreibzugriff auf das Instanzverzeichnis. Wenn die Verzeichnisse in einem anderen Pfad als **DB2PATH** erstellt werden sollen, müssen Sie die Variable **DB2INSTPROF** definieren, bevor Sie den Befehl **db2icrt** eingeben.
Für DB2 Enterprise Server Edition (ESE) müssen Sie außerdem deklarieren, dass Sie eine neue Instanz hinzufügen, bei der es sich um ein partitioniertes Datenbanksystem handelt. Wenn Sie darüber hinaus mit einer ESE-Instanz arbeiten, die mehr als eine Datenbankpartition hat, und Fast Communication Manager (FCM) verwenden, können Sie mehrere Verbindungen zwischen Datenbankpartitionen haben, indem Sie bei der Erstellung der Instanz mehrere TCP/IP-Ports definieren.
Beispiel: Verwenden Sie bei Windows-Betriebssystemen den Befehl **db2icrt** mit dem Parameter **-r** *portbereich*. Der Portbereich wird wie folgt dargestellt, wobei *basisport* der erste Port und *endport* der letzte Port in einem Bereich von Portnummern ist, die von FCM verwendet werden können:

```
-r:basisport,endport
```

Ändern von Instanzen

Die Instanzen wurden so gestaltet, dass sie durch ein nachfolgendes Installieren und Deinstallieren anderer Produkte möglichst wenig beeinflusst werden. Unter Linux und UNIX können Sie Instanzen nach der Installation oder dem Entfernen von ausführbaren Dateien oder Komponenten aktualisieren. Unter Windows führen Sie den Befehl **db2iupdt** aus.

In den meisten Fällen erhalten bzw. verlieren vorhandene Instanzen automatisch den Zugriff auf die Funktionen des Produkts, das installiert bzw. deinstalliert wird. Beim Installieren bestimmter Programmdateien oder Komponenten übernehmen die vorhandenen Instanzen jedoch nicht automatisch die neuen Systemkonfigurationsparameter oder erhalten nicht automatisch Zugriff auf die neu hinzugekommenen Funktionen. In solchen Fällen muss die betreffende Instanz aktualisiert werden.

Wenn der Datenbankmanager durch Installieren einer vorläufigen Programmkorrektur (PTF - Program Temporary Fix) oder eines Patch-Codes aktualisiert wird, sollten alle vorhandenen Datenbankinstanzen mithilfe des Befehls **db2iupdt** (Rootinstallationen) oder des Befehls **db2nrupdt** (Nichtrootinstallationen) aktualisiert werden.

Machen Sie sich zunächst mit den vorhandenen Instanzen und Datenbankpartitionsservern vertraut, bevor Sie versuchen, eine Instanz zu ändern oder zu löschen.

Aktualisieren der Instanzkonfiguration (Linux, UNIX)

Zum Aktualisieren der Konfiguration für Rootinstanzen unter Linux- und UNIX-Betriebssystemen verwenden Sie den Befehl **db2iupdt**. Zur Aktualisierung von Nichtrootinstanzen führen Sie den Befehl **db2nrupdt** aus.

Informationen zu diesem Vorgang

Der Befehl **db2iupdt** aktualisiert die angegebene Instanz durch folgende Operationen:

- Die Dateien im Unterverzeichnis `sqllib` im Ausgangsverzeichnis des Instanzeigners werden ersetzt.
- Wenn der Knotentyp geändert wurde, wird eine neue Konfigurationsdatei für den Datenbankmanager erstellt. Dabei werden Werte aus der vorhandenen Konfigurationsdatei des Datenbankmanagers mit der Standardkonfigurationsdatei des Datenbankmanagers für den neuen Knotentyp zusammengefügt. Beim Erstellen einer neuen Konfigurationsdatei für den Datenbankmanager wird die alte Datei im Unterverzeichnis `backup` des Unterverzeichnisses `sqllib` im Ausgangsverzeichnis des Instanzeigners gesichert.

Der Befehl **db2iupdt** befindet sich im Verzeichnis *DB2DIR*`/instance`, wobei *DB2DIR* die Position angibt, in der die aktuelle Version des DB2-Datenbankprodukts installiert ist.

Einschränkungen

Diese Task kann nur auf Rootinstanzen angewendet werden.

Vorgehensweise

Geben Sie Folgendes ein, um eine Instanz über die Befehlszeile zu aktualisieren:

```
db2iupdt InstName
```

Die Angabe *InstName* ist der Anmeldename des Instanzeigners.
Für diesen Befehl sind weitere optionale Parameter verfügbar:

-h oder -?
Zeigt ein Hilfemenü für diesen Befehl an.

-d
Dieser Parameter konfiguriert den Debugmodus für die Fehlerbestimmung.

-a *AuthTyp*
Dieser Parameter gibt den Authentifizierungstyp für die Instanz an. Gültige Authentifizierungstypen sind `SERVER`, `SERVER_ENCRYPT` und `CLIENT`. Wenn dieser Parameter nicht angegeben wird, wird standardmäßig der Authentifizierungstyp `SERVER` verwendet, wenn ein DB2-Server installiert ist. Ansonsten wird die Option `CLIENT` verwendet. Der für die Instanz angegebene Authentifizierungstyp gilt für alle Datenbanken, die zu dieser Instanz gehören.

-e
Ermöglicht Ihnen die Aktualisierung aller vorhandenen Instanzen. Mit dem Befehl **db2ilist** können Sie die vorhandenen Instanzen auflisten.

-u *AbgeschirmteID*
Dieser Parameter gibt den Benutzer an, unter dem die abgeschirmten benutzerdefinierten Funktionen (UDFs) und die gespeicherten Prozeduren ausgeführt werden. Dieser Parameter ist nicht erforderlich, wenn Sie den Data Server Cli-

ent oder das DB2 Software Developer's Kit installieren. Für andere DB2-Datenbankprodukte ist dies ein erforderlicher Parameter. Anmerkung: *Abgeschirmte_ID* darf nicht "root" oder "bin" sein.

-k Dieser Parameter behält den aktuellen Instanztyp bei. Wenn Sie diesen Parameter nicht angeben, wird die aktuelle Instanz in der nachfolgend angegebenen Reihenfolge auf den höchsten verfügbaren Instanztyp hochgestuft:

- Partitionierter Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen und fernen Clients
- Client

Beispiel

- Wenn Sie nach dem Erstellen der Instanz DB2 Workgroup Server Edition oder DB2 Enterprise Server Edition installiert haben, geben Sie den folgenden Befehl ein, um die Instanz zu aktualisieren:

  ```
  db2iupdt -u db2fenc1 db2inst1
  ```

- Wenn Sie nach dem Erstellen der Instanz DB2 Connect Enterprise Edition installiert haben, können Sie den Instanznamen wie folgt auch als abgeschirmte ID verwenden:

  ```
  db2iupdt -u db2inst1 db2inst1
  ```

- Zur Aktualisierung von Clientinstanzen rufen Sie den folgenden Befehl auf:

  ```
  db2iupdt db2inst1
  ```

Aktualisieren der Instanzkonfiguration (Windows)

Zur Aktualisierung der Instanzkonfiguration unter Windows verwenden Sie den Befehl **db2iupdt**.

Informationen zu diesem Vorgang

Der Befehl **db2iupdt** aktualisiert die angegebene Instanz durch folgende Operationen:

- Die Dateien im Unterverzeichnis `sqllib` im Ausgangsverzeichnis des Instanzeigners werden ersetzt.
- Wenn der Knotentyp geändert wird, wird eine neue Konfigurationsdatei für den Datenbankmanager erstellt. Dabei werden Werte aus der vorhandenen Konfigurationsdatei des Datenbankmanagers mit der Standardkonfigurationsdatei des Datenbankmanagers für den neuen Knotentyp zusammengefügt. Beim Erstellen einer neuen Konfigurationsdatei für den Datenbankmanager wird die alte Datei im Unterverzeichnis `backup` des Unterverzeichnisses `sqllib` im Ausgangsverzeichnis des Instanzeigners gesichert.

Der Befehl **db2iupdt** befindet sich im Verzeichnis `\sqllib\bin`.

Vorgehensweise

Zur Aktualisierung der Instanzkonfiguration verwenden Sie den Befehl **db2iupdt**. Beispiel:

```
db2iupdt InstName
```

Die Angabe *InstName* ist der Anmeldename des Instanzeigners.
Für diesen Befehl sind weitere optionale Parameter verfügbar:

/h: *hostname*
Überschreibt den Standard-TCP/IP-Hostnamen, wenn es einen oder mehrere TCP/IP-Hostnamen für den aktuellen Computer gibt.

/p: *instanzprofilpfad*
Gibt den neuen Instanzprofilpfad für die aktualisierte Instanz an.

/r: *basisport*,*endport*
Gibt den Bereich der TCP/IP-Ports an, die von der partitionierten Datenbankinstanz beim Betrieb mit mehreren Datenbankpartitionen verwendet werden.

/u: *benutzername*,*kennwort*
Gibt den Kontonamen und das Kennwort für den DB2-Service (Dienst) an.

Arbeiten mit Instanzen

Bei der Arbeit mit Instanzen können Sie Instanzen starten oder stoppen sowie Verbindungen zu Instanzen herstellen (ATTACH) oder trennen (DETACH).

Informationen zu diesem Vorgang

Jede Instanz wird von Benutzern verwaltet, die zur Gruppe **sysadm_group** gehören, die in der *Instanzkonfigurationsdatei* definiert ist. Diese Datei wird auch als *Konfigurationsdatei des Datenbankmanagers* bezeichnet. Die Erstellung von Benutzer-IDs und Benutzergruppen ist je nach Betriebsumgebung unterschiedlich.

Automatisches Starten von Instanzen

Sie können Instanzen so aktivieren, dass diese nach jedem Systemwiederanlauf automatisch gestartet werden. Die zum Ausführen dieser Task erforderlichen Schritte sind je nach Betriebssystem unterschiedlich.

Informationen zu diesem Vorgang

Unter Windows-Betriebssystemen wird die bei der Installation erstellte Datenbankinstanz standardmäßig auf automatischen Start eingestellt.

Unter Linux, UNIX- und Windows-Betriebssystemen ist eine mithilfe von **db2icrt** erstellte Instanz auf automatischen Start eingestellt.

Vorgehensweise

Gehen Sie wie folgt vor, um eine Instanz für den automatischen Start zu konfigurieren:

- Unter Windows-Betriebssystemen müssen Sie zum Fenster **Dienste** wechseln und dort die Eigenschaft des DB2-Service ändern.
- Geben Sie unter Linux- und UNIX-Betriebssystemen den folgenden Befehl ein:
  ```
  db2iauto -on instanzname
  ```
  Dabei ist *instanzname* der Anmeldename der Instanz.

Starten von Instanzen (Linux, UNIX)

Sie müssen eine DB2-Datenbank während normaler Geschäftsoperationen möglicherweise starten oder stoppen. Sie müssen beispielsweise eine Instanz starten, um einige der folgenden Tasks ausführen zu können: Verbindung zu einer Datenbank in der Instanz herstellen, eine Anwendung vorkompilieren, ein Paket an eine Datenbank binden oder auf Hostdatenbanken zugreifen.

Vorbereitende Schritte

Führen Sie folgende Schritte aus, bevor Sie eine Instanz unter Ihrem Linux- oder UNIX-Betriebssystem starten:

1. Melden Sie sich mit einer Benutzer-ID oder einem Benutzernamen an, die bzw. der über die Berechtigung SYSADM, SYSCTRL oder SYSMAINT für die Instanz verfügt. Oder melden Sie sich als Instanzeigner an.
2. Führen Sie das Startscript wie folgt aus, wobei *INSTHOME* das Ausgangsverzeichnis der Instanz ist, die Sie verwenden wollen:

```
. INSTHOME/sqllib/db2profile      (für Bourne- oder Korn-Shell)
source INSTHOME/sqllib/db2cshrc   (für C-Shell)
```

Vorgehensweise

Gehen Sie wie folgt vor, um die Instanz zu starten:

- Geben Sie den Befehl **db2start** in die Befehlszeile ein. Der DB2-Datenbankmanager wendet den Befehl auf die aktuelle Datenbank an.
- Öffnen Sie in IBM Data Studio den Taskassistenten zum Starten der Instanz.

Zugehörige Informationen:

Starten von Instanzen (Windows)

Sie müssen eine DB2-Instanz während normaler Geschäftsoperationen möglicherweise starten oder stoppen. Sie müssen beispielsweise eine Instanz starten, um einige der folgenden Tasks ausführen zu können: Verbindung zu einer Datenbank in der Instanz herstellen, eine Anwendung vorkompilieren, ein Paket an eine Datenbank binden oder auf eine Hostdatenbank zugreifen.

Vorbereitende Schritte

Damit die DB2-Datenbankinstanz erfolgreich als Service (bzw. Dienst) gestartet werden kann, muss das Benutzerkonto über das richtige Zugriffsrecht verfügen, wie es vom Betriebssystem Windows zum Starten eines Windows-Dienstes definiert wird. Das Benutzerkonto kann Mitglied der Gruppe der Administratoren, der Serveroperatoren oder der Hauptbenutzer sein. Wenn die erweiterte Sicherheit aktiviert ist, können nur Mitglieder der Gruppe DB2ADMNS und der Administratorgruppe die Datenbank standardmäßig starten.

Informationen zu diesem Vorgang

Durch den Befehl **db2start** wird die DB2-Datenbankinstanz standardmäßig als Windows-Dienst (Service) gestartet. Die DB2-Datenbankinstanz unter Windows kann immer noch als Prozess ausgeführt werden, indem der Parameter **/D** für den Befehl **db2start** angegeben wird. Die DB2-Datenbankinstanz kann als Service auch über die **Systemsteuerung** oder mithilfe des Befehls **NET START** gestartet werden.

Bei der Ausführung in einer Umgebung mit partitionierten Datenbanken wird jeder Datenbankpartitionsserver als Windows-Dienst gestartet. In einer Umgebung mit partitionierten Datenbanken können Sie den Parameter **/D** nicht verwenden, um eine DB2-Instanz als Prozess zu starten.

Vorgehensweise

Gehen Sie wie folgt vor, um die Instanz zu starten:

- Geben Sie den Befehl **db2start** in die Befehlszeile ein. Der DB2-Datenbankmanager wendet den Befehl auf die aktuelle Datenbank an.
- Öffnen Sie in IBM Data Studio den Taskassistenten zum Starten der Instanz.

Zugehörige Informationen:

Herstellen und Trennen von Verbindungen zu Instanzen (mit ATTACH und DETACH)

Zum Herstellen der Verbindung zu einer anderen Instanz der Datenbank, die auch eine ferne Instanz sein kann, verwenden Sie auf allen Plattformen den Befehl **ATTACH**. Zum Trennen der Verbindung zu einer Instanz verwenden Sie den Befehl **DETACH**.

Vorbereitende Schritte

Es muss mehr als eine Instanz vorhanden sein.

Vorgehensweise

- Gehen Sie wie folgt vor, um eine Zuordnung zu einer Instanz herzustellen:
  - Geben Sie den Befehl **ATTACH** in die Befehlszeile ein:
  - Rufen Sie aus einer Clientanwendung heraus die API sqleatin auf.
- Gehen Sie wie folgt vor, um die Zuordnung zu einer Instanz aufzuheben:
  - Geben Sie den Befehl **DETACH** in die Befehlszeile ein.
  - Rufen Sie aus einer Clientanwendung heraus die API sqledtin auf.

Beispiel

Um zum Beispiel eine Verbindung zu einer Instanz mit dem Namen 'testdb2' herzustellen, die zuvor im Knotenverzeichnis katalogisiert wurde, müssen Sie folgenden Befehl eingeben:

```
db2 attach to testdb2
```

Heben Sie die Zuordnung zu einer Instanz nach Ausführen der Wartungsaktivitäten für die Instanz 'testdb2' auf:

```
db2 detach
```

Arbeiten mit Instanzen in derselben oder in anderen DB2-Kopien

Sie können mehrere Instanzen gleichzeitig in derselben DB2-Kopie oder in verschiedenen DB2-Kopien ausführen.

Informationen zu diesem Vorgang

Damit eine Instanz nicht auf die Datenbanken einer anderen Instanz zugreift, werden die Datenbankdateien für eine Instanz unter einem Verzeichnis erstellt, das den gleichen Namen wie die Instanz besitzt. Wenn zum Beispiel eine Datenbank auf dem Laufwerk `C:` für die Instanz `DB2` erstellt wird, werden die Datenbankdateien in einem Verzeichnis mit dem Namen `C:\DB2` erstellt. Analog werden bei der Erstellung einer Datenbank auf Laufwerk `C:` für die Instanz `TEST` die Datenbankdateien in einem Verzeichnis mit dem Namen `C:\TEST` erstellt. Standardmäßig entspricht der Wert dem Buchstaben des Laufwerks, in dem das DB2-Produkt installiert ist. Weitere Informationen finden Sie im Abschnitt zum Datenbankmanagerkonfigurationsparameter **dftdbpath**.

Vorgehensweise

- Wenn Sie mit Instanzen in derselben DB2-Kopie arbeiten möchten, müssen Sie wie folgt vorgehen:
  1. Führen Sie das Erstellen bzw. das Upgrade aller Instanzen in ein und derselben DB2-Kopie aus.
  2. Setzen Sie die Umgebungsvariable **DB2INSTANCE** auf den Namen der Instanz, mit der Sie arbeiten. Diese Aktion muss auftreten, bevor Sie Befehle für die Datenbank absetzen.
- Verwenden Sie eine der folgenden Methoden, wenn Sie mit einer Instanz in einem System mit mehreren DB2-Kopien arbeiten möchten:
  - Verwenden Sie das Befehlsfenster über **Start** > **Programme** > **IBM DB2** > ***name_der_db2-kopie*** > **Befehlszeilentools** > **Befehlsfenster**. Das Befehlsfenster ist bereits mit den richtigen Umgebungsvariablen für die jeweils ausgewählte DB2-Kopie eingerichtet.
  - Verwenden Sie `db2envar.bat` über ein Befehlsfenster:
    1. Öffnen Sie ein Befehlsfenster.
    2. Führen Sie die Datei `db2envar.bat` aus und verwenden Sie dabei den vollständig qualifizierten Pfad für die DB2-Kopie, die die Anwendung verwenden soll:

       installationsverz_der_db2-kopie`\bin\db2envar.bat`

Stoppen von Instanzen (Linux, UNIX)

Möglicherweise müssen Sie die aktuelle Instanz des Datenbankmanagers stoppen.

Vorbereitende Schritte

1. Melden Sie sich mit einer Benutzer-ID oder einem Benutzernamen, die bzw. der über die Berechtigung SYSADM, SYSCTRL oder SYSMAINT verfügt, bei der Instanz an bzw. stellen Sie die Verbindung (ATTACH) zu der Instanz her. Oder melden Sie sich als Instanzeigner an.
2. Zeigen Sie alle Anwendungen und Benutzer an, die mit der Datenbank, die Sie stoppen wollen, verbunden sind. Zeigen Sie die Anwendungen als Liste an, um sicherzustellen, dass keine wichtigen oder kritischen Anwendungen ausgeführt werden. Sie benötigen für diese Aktivität die Berechtigung SYSADM, SYSCTRL oder SYSMAINT.
3. Erzwingen Sie die Trennung aller Anwendungen und Benutzer von der Datenbank. Für das Erzwingen der Trennung von Benutzern benötigen Sie die Berechtigung SYSADM oder SYSCTRL.
4. Wenn Sitzungen des Befehlszeilenprozessors mit einer Instanz verbunden sind, müssen Sie jede dieser Sitzungen einzeln durch Ausführen des Befehls **TERMINATE** beenden, bevor Sie den Befehl **db2stop** ausführen.

Informationen zu diesem Vorgang

Wenn Sie Befehle zum Starten oder Stoppen einer Instanz ausführen, wendet der DB2-Datenbankmanager den Befehl auf die aktuelle Instanz an. Weitere Informationen finden Sie in „Ermitteln der aktuellen Instanz” auf Seite 624.

Einschränkungen

Der Befehl **db2stop** kann nur auf dem Server ausgeführt werden.

Bei der Ausführung dieses Befehls sind keine Datenbankverbindungen zulässig. Wenn es Instanzverbindungen (mit ATTACH) gibt, werden diese zwangsweise getrennt, bevor die Instanz gestoppt wird.

Vorgehensweise

Gehen Sie wie folgt vor, um eine Instanz unter einem Linux- oder UNIX-Betriebssystem zu stoppen:

- Geben Sie den Befehl **db2stop** in die Befehlszeile ein. Der DB2-Datenbankmanager wendet den Befehl auf die aktuelle Datenbank an.
- Öffnen Sie in IBM Data Studio den Taskassistenten zum Stoppen der Instanz.

Zugehörige Informationen:

Stoppen von Instanzen (Windows)

Möglicherweise müssen Sie die aktuelle Instanz des Datenbankmanagers stoppen.

Vorbereitende Schritte

1. Das Benutzerkonto, das den DB2-Datenbankservice stoppt, muss über das richtige im Windows-Betriebssystem definierte Zugriffsrecht verfügen. Das Benutzerkonto kann Mitglied der Gruppe der Administratoren, der Serveroperatoren oder der Hauptbenutzer sein.
2. Zeigen Sie alle Anwendungen und Benutzer an, die mit der Datenbank, die Sie stoppen wollen, verbunden sind. Zeigen Sie die Anwendungen als Liste an, um sicherzustellen, dass keine wichtigen oder kritischen Anwendungen ausgeführt werden. Sie benötigen für diese Aktivität die Berechtigung SYSADM, SYSCTRL oder SYSMAINT.
3. Erzwingen Sie die Trennung aller Anwendungen und Benutzer von der Datenbank. Für das Erzwingen der Trennung von Benutzern benötigen Sie die Berechtigung SYSADM oder SYSCTRL.
4. Wenn Sitzungen des Befehlszeilenprozessors mit einer Instanz verbunden sind, müssen Sie jede dieser Sitzungen einzeln durch Ausführen des Befehls **TERMINATE** beenden, bevor Sie den Befehl **db2stop** ausführen.

Informationen zu diesem Vorgang

Anmerkung: Wenn Sie Befehle zum Starten oder Stoppen einer Instanz ausführen, wendet der Datenbankmanager den Befehl auf die aktuelle Instanz an. Weitere Informationen zu diesem Thema finden Sie in „Ermitteln der aktuellen Instanz” auf Seite 624.

Einschränkungen

Der Befehl **db2stop** kann nur auf dem Server ausgeführt werden.

Bei der Ausführung dieses Befehls sind keine Datenbankverbindungen zulässig. Wenn jedoch Instanzverbindungen vorhanden sind, werden diese zwangsweise getrennt, bevor der DB2-Datenbankservice gestoppt wird.

Wenn Sie den Datenbankmanager in einer Umgebung mit partitionierten Datenbanken verwenden, wird jeder Datenbankpartitionsserver als Service gestartet. Jeder dieser Services muss gestoppt werden.

Vorgehensweise

Gehen Sie wie folgt vor, um die Instanz zu stoppen:

- Geben Sie den Befehl **db2start** in der Befehlszeile ein. Der DB2-Datenbankmanager wendet den Befehl auf die aktuelle Instanz an.
- Geben Sie den Befehl **NET STOP** in die Befehlszeile ein.
- Öffnen Sie in IBM Data Studio den Taskassistenten zum Stoppen der Instanz.

Zugehörige Informationen:

Löschen von Instanzen

Führen Sie zum Löschen einer Rootinstanz den Befehl **db2idrop** aus. Zum Löschen von Nichtrootinstanzen müssen Sie Ihr DB2-Datenbankprodukt deinstallieren.

Vorgehensweise

Gehen Sie wie folgt vor, um eine Rootinstanz über die Befehlszeile zu entfernen:

1. Stoppen Sie alle Anwendungen, die momentan mit der Instanz arbeiten.
2. Stoppen Sie den Befehlszeilenprozessor durch Ausführen des Befehls **terminate** in jedem Befehlsfenster.
3. Stoppen Sie die Instanz durch Ausführen des Befehls **db2stop**.
4. Führen Sie ein Backup für das in der Registrierdatenbankvariablen **DB2INSTPROF** angegebene Instanzverzeichnis durch.

   Unter Linux- und UNIX-Betriebssystemen könnte es sinnvoll sein, die Dateien im Verzeichnis *INSTHOME*/sqllib durch ein Backup zu sichern (dabei ist *INSTHOME* das Ausgangsverzeichnis des Instanzeigners). Beispielsweise könnten Sie beabsichtigen, die Konfigurationsdatei des Datenbankmanagers (db2systm), die Datei db2nodes.cfg, die benutzerdefinierten Funktionen oder die abgeschirmten gespeicherten Prozeduren zu sichern.
5. Nur Linux- und UNIX-Betriebssysteme: Melden Sie sich als Instanzeigner ab und als Benutzer mit Rootberechtigung an.
6. Setzen Sie den Befehl **db2idrop** ab. Zum Beispiel:

   ```
   db2idrop InstName
   ```

   Dabei ist *InstName* der Name der zu löschenden Instanz.

   Der Befehl **db2idrop** entfernt den Instanzeintrag aus der Instanzliste und löscht das Unterverzeichnis sqllib im Ausgangsverzeichnis des Instanzeigners.

   Anmerkung: Wenn Sie unter Linux- und UNIX-Betriebssystemen den Befehl **db2idrop** ausführen und eine Nachricht empfangen, die besagt, dass das Unterverzeichnis *INSTHOME*/sqllib nicht entfernt werden kann, könnte eine Ursache dafür sein, dass das Unterverzeichnis *INSTHOME*/adm Dateien mit der Erweiterung .nfs enthält. Das Unterverzeichnis adm ist ein über NFS angehängtes System, und die Dateien werden auf dem Server gesteuert. Sie müssen die Dateien *.nfs von der Position auf dem Dateiserver löschen, an der das Verzeichnis angehängt ist. Anschließend können Sie das Unterverzeichnis *INSTHOME*/sqllib entfernen.
7. Wenn unter Windows-Betriebssystemen die Instanz, die Sie gelöscht haben, die Standardinstanz war, legen Sie eine neue Standardinstanz fest, indem Sie den Befehl **db2set** ausführen:

   ```
   db2set db2instdef=instanzname -g
   ```

Dabei ist *instanzname* der Name einer vorhandenen Instanz.

8. Linux- und UNIX-Betriebssysteme: Entfernen Sie die Benutzer-ID und Gruppe des Instanzeigner (wenn sie nur für diese Instanz verwendet wird). Löschen Sie die Benutzer-ID und die Gruppe nicht, wenn Sie vorhaben, die Instanz erneut zu erstellen.

   Dieser Schritt ist optional, da die Benutzer-ID und die Gruppe des Instanzeigners möglicherweise auch für andere Zwecke verwendet werden.

Instanzverwaltung in einer DB2 pureScale-Umgebung

Dieser Abschnitt enthält Informationen über die Verwaltung einer DB2 pureScale-Instanz. Die hier behandelten Themen beziehen sich speziell auf DB2 pureScale Feature und bieten keine allgemeineren Verwaltungsinformationen zum DB2-Datenbankprodukt.

Dieser Abschnitt enthält Informationen über grundlegende Verwaltungskonzepte und -tasks sowie Benutzerszenarios für die folgenden Bereiche:

- Starten und Stoppen von Membern und Cluster-Caching-Funktionen
- Wartungsaufgaben wie das Aktualisieren von Cluster-Caching-Funktionen oder Member-Hosts und das Hinzufügen von Ressourcen zu dem gemeinsam genutzten Dateisystemcluster.
- Automatisches Neustarten ausgefallener Member und Cluster-Caching-Funktionen
- Konfigurieren des Clusters für gemeinsam genutzte Dateisysteme, der Cluster-Caching-Funktionen und der Pufferpools

Mehrere aktive Datenbanken in einer DB2 pureScale-Umgebung

Seit DB2 Version 9.8 Fixpack 3 können Sie nun mit mehreren aktiven Datenbanken in einer DB2 pureScale-Umgebung arbeiten.

In bisherigen Iterationen von DB2 Version 9.8 unterschied sich die DB2 pureScale-Umgebung von der DB2 Enterprise Server Edition-Umgebung und der partitionierten Datenbankumgebung in dem Maße, als dass immer nur eine Datenbank aktiv sein konnte. Seit DB2 Version 9.8 Fixpack 3 gibt es diese Einschränkung nicht mehr.

Die Funktionalität für den Benutzer in einer DB2 pureScale-Umgebung entspricht nun fast vollständig der Funktionalität mit mehreren aktiven Datenbanken in DB2 Enterprise Server Edition-Umgebungen und partitionierten Datenbankumgebungen.

Außerhalb einer DB2 pureScale-Umgebung können maximal 255 Datenbanken zu einem bestimmten Zeitpunkt aktiv sein. Innerhalb einer DB2 pureScale-Umgebung ist dieser Wert jedoch 200.

Die standardmäßig verwendete maximale Anzahl aktiver Datenbanken in einer DB2 pureScale-Umgebung ist 32. Dies ist zudem der neue Standardwert für DB2 Enterprise Server Edition-Umgebungen und partitionierte Datenbankumgebungen.

Informationen darüber, wie Sie mehrere aktive Datenbanken in einer DB2 pureScale-Umgebung umsetzen können, finden Sie im Abschnitt zu den Konfigurationsparameteränderungen unter „Konfiguration der CF-Speicherparameter für DB2 pureScale” auf Seite 980.

Um die Anzahl der aktiven Datenbanken in einer DB2 pureScale-Umgebung zu ändern, können Sie den Konfigurationsparameter **numdb** modifizieren. Die Änderungen werden dann nach dem nächsten globalen Neustart wirksam.

Zusätzlich zu dem Grenzwert **numdb** besteht in einer DB2 pureScale-Umgebung ebenfalls ein maximaler Wert für die Anzahl der Datenbankaktivierungen für alle Member in einer Instanz. Die maximale Anzahl beträgt 512 Datenbankaktivierungen. Beispiel: Wenn jedes Member in einer DB2 pureScale-Umgebung mit vier Membern 200 Datenbanken aktiviert, bedeutet dies eine Gesamtzahl von 800 Datenbankaktivierungen durch Member. Da die Zahl von 800 Datenbankaktivierungen den maximalen Wert überschreitet, wird ein Fehler zurückgegeben.

Wenn in Umgebungen mit mehreren Datenbanken nach dem Absturz eines Members eine Recovery erforderlich wird, wird die Anzahl der Datenbanken, die auf jedem Member parallel zueinander wiederhergestellt werden, durch den Wert des Konfigurationsparameters **numdb** oder der Registrierdatenbankvariable **DB2_MCR_RECOVERY_PARALLELISM_CAP** festgelegt, je nachdem, welcher dieser beiden Werte kleiner ist.

Starten und Stoppen von Clusterkomponenten und Datenbanken in einer DB2 pureScale-Umgebung

In einer DB2 pureScale-Umgebung erfolgt das Starten und Stoppen einer Cluster-Caching-Funktion oder eines Members im Rahmen eines globalen Befehls **db2start** bzw. **db2stop**.

Wenn Sie den Befehl **db2start** oder **db2stop** ausgeben, werden alle zu diesem Zeitpunkt definierten Member und Cluster-Caching-Funktionen gestartet bzw. gestoppt. In manchen Fällen kann es aber auch sinnvoll sein, diese Clusterkomponenten und Datenbanken differenzierter zu starten und zu stoppen.

Starten einer Cluster-Caching-Funktion

Eine Cluster-Caching-Funktion (CF) wird durch einen global ausgeführten Befehl **db2start** oder durch einen individuell ausgeführten Befehl **db2start** CF gestartet. Diese Task behandelt in erster Linie das Starten einer einzelnen Cluster-Caching-Funktion.

Informationen zu diesem Vorgang

Bei einem global ausgeführten Befehl **db2start** versucht der Cluster-Manager, die primäre Rolle (Status PRIMARY) auf der bevorzugten primären Cluster-Caching-Funktion zu starten. Die andere Cluster-Caching-Funktion wird in der sekundären Rolle (Status PEER) gestartet.

Wenn bereits eine Cluster-Caching-Funktion gestartet ist und im Status PRIMARY ausgeführt wird, tritt die nächste Cluster-Caching-Funktion, die Sie starten (die sekundäre Cluster-Caching-Funktion), in eine CATCHUP-Phase ein, die sicherstellt, dass die sekundäre Cluster-Caching-Funktion eine Kopie aller relevanten Informationen der primären Cluster-Caching-Funktion in ihrem Speicher (d. h. im Speicher der sekundären Cluster-Caching-Funktion) hat, bevor sie in den Status PEER übergeht.

Vorgehensweise

Gehen Sie wie folgt vor, um eine bestimmte Cluster-Caching-Funktion zu starten:

Setzen Sie den folgenden Befehl ab:

```
db2start CF cf-id
```

Beispiel

John, ein Datenbankadministrator, hat eine zweite Cluster-Caching-Funktion zu einer DB2 pureScale-Instanz hinzugefügt beschrieben. An diesem Punkt fragt er den Status aller Cluster-Caching-Funktionen in der Instanz mit der folgenden Abfrage ab:

```
SELECT ID,
       varchar(CURRENT_HOST,10) AS CUR_HOST,
       varchar(STATE,17) AS STATE,
       ALERT
FROM SYSIBMADM.DB2_CF
```

Und erhält die folgende Ausgabe:

```
ID     CUR_HOST     STATE      ALERT
------ ------------ ---------- -----
   128 so5          PRIMARY    NO
   129 so6          STOPPED    NO

     2 Satz/Sätze ausgewählt.
```

Die Cluster-Caching-Funktion mit der ID 128 ist die einzige aktive Cluster-Caching-Funktion und befindet sich daher auch im Status PRIMARY.

John führt den Befehl `db2start CF 129` aus und fragt den Status der Cluster-Caching-Funktionen mit der folgenden Abfrage ab:

```
SELECT ID,
       varchar(CURRENT_HOST,10) AS CUR_HOST,
       varchar(STATE,17) AS STATE,
       ALERT
FROM SYSIBMADM.DB2_CF
```

Und erhält die folgende Ausgabe:

```
ID     CUR_HOST     STATE        ALERT
------ ------------ ----------   -----
   128 so5          PRIMARY      NO
   129 so6          CATCHUP(50%) NO

     2 Satz/Sätze ausgewählt.
```

Die Cluster-Caching-Funktion mit der ID 129 empfängt nun eine Kopie aller relevanten Informationen von der Cluster-Caching-Funktion mit der ID 128, sodass sie in die Lage versetzt wird, die Rolle als primäre Cluster-Caching-Funktion zu übernehmen, wenn die Cluster-Caching-Funktion mit der ID 128 ausfällt.

Der Datenbankadministrator fragt den Status aller Cluster-Caching-Funktionen erneut mit der folgenden Abfrage ab:

```
SELECT ID,
       varchar(CURRENT_HOST,10) AS CUR_HOST,
       varchar(STATE,17) AS STATE,
       ALERT
FROM SYSIBMADM.DB2_CF
```

Und erhält die folgende Ausgabe:

```
ID     CUR_HOST     STATE      ALERT
------ ------------ ---------- -----
   128 so5          PRIMARY    NO
   129 so6          PEER       NO

     2 Satz/Sätze ausgewählt.
```

Jetzt befindet sich die Cluster-Caching-Funktion mit der ID 129 im Status PEER und ist bereit, die Rolle als primäre Cluster-Caching-Funktion zu übernehmen, wenn die aktuelle primäre Cluster-Caching-Funktion ausfällt.

Stoppen einer Cluster-Caching-Funktion

Eine Cluster-Caching-Funktion (CF) wird durch einen global ausgeführten Befehl **db2stop** oder durch einen individuell ausgeführten Befehl **db2stop CF** gestoppt. Dieser Abschnitt behandelt in erster Linie das Stoppen einer einzelnen Cluster-Caching-Funktion.

Informationen zu diesem Vorgang

Setzen Sie den Befehl **db2stop CF** ab, wenn Sie eine Cluster-Caching-Funktion auf einem Host stoppen, jedoch gleichzeitig andere Instanzprozesse auf demselben Host weiterhin ausführen möchten. Sie können den Befehl **db2stop CF** vor dem Beenden eines Hosts verwenden.

Einschränkungen

Sie können jedoch nicht die primäre Cluster-Caching-Funktion stoppen, wenn die folgenden Situationen zutreffen:

- Es befinden sich aktive Member in der Instanz.
- Die primäre Cluster-Caching-Funktion enthält genutzte Seiten.
- Die primäre Cluster-Caching-Funktion enthält Sperren.
- Die sekundäre Cluster-Caching-Funktion befindet sich nicht im PEER-Status.

Sie können jedoch die sekundäre Cluster-Caching-Funktion mit der Option **FORCE** stoppen, auch wenn eine dieser Situationen vorliegt.

Vorgehensweise

Setzen Sie zum Stoppen einer bestimmten Cluster-Caching-Funktion den folgenden Befehl ab:

```
db2stop CF cf-id
```

Starten eines Members

DB2-Member werden durch einen global ausgeführten Befehl **db2start** oder einen individuell ausgeführten Befehl **db2start** gestartet. Dieser Abschnitt behandelt in erster Linie das Starten eines einzelnen Members.

Informationen zu diesem Vorgang

Wenn Sie einen Befehl **db2start** (entweder für die Instanz oder das Member) absetzen, startet der Datenbankmanager alle inaktiven DB2-Prozesse auf dem Host und alle in der Instanz (mit aktiven Hosts) definierten Cluster-Caching-Funktionen, falls die inaktiven Prozesse und die Cluster-Caching-Funktionen nicht bereits ausgeführt werden. Wenn die Cluster-Caching-Funktionen nicht gestartet werden können, schlägt die Memberstartoperation fehl. Wenn die Leerlaufprozesse auf dem

Benutzerhost des Members nicht gestartet werden können, schlägt die Startoperation fehl, jedoch wird das Member von den DB2-Cluster-Services auf einem anderen Host im Light-Neustartmodus gestartet. (Weitere Informationen finden Sie in Light-Neustart.)

Vorgehensweise

Setzen Sie zum Starten eines bestimmten Members den folgenden Befehl ab:

```
db2start member member-id
```

Ergebnisse

Der Datenbankmanager startet einzelne Member auf dem Host, der gegenwärtig in der Datei 'db2nodes.cfg' angegeben ist, auch wenn dieser Host nicht der Benutzerhost für das im Befehl angegebene Member ist. Mit anderen Worten, ein Member, das zuvor als Gastmember auf einem anderen Host ausgeführt wurde, wird auf diesem Host im Light-Neustartmodus gestartet. Wenn der Benutzerhost dieses Members aktiv und bereit ist, sein residentes Member zu empfangen, übertragen die DB2-Cluster-Services das Member auf seinen Benutzerhost zurück (Failback).

Beispiel

Der Datenbankadministrator John hat Wartungsarbeiten an einem Host beendet, auf dem nur ein Member (Member 0) definiert ist. Er hat jetzt die Instanz auf dem Host erneut gestartet (`db2start instance on host so1`). Das Member wurde vor den Wartungsarbeiten beendet. Der Datenbankadministrator fragt den Status aller Member in der Instanz mit der folgenden Abfrage ab:

```
SELECT ID,
       varchar(HOME_HOST,10) AS HOME_HOST,
       varchar(CURRENT_HOST,10) AS CUR_HOST,
       varchar(STATE,21) AS STATE,
       ALERT
FROM SYSIBMADM.DB2_MEMBER
```

Und er erhält die folgende Ausgabe:

```
ID     HOME_HOST CUR_HOST     STATE      ALERT
------ --------- ------------ ---------- -----
     0 so1       so1          STOPPED    NO
     2 so2       so2          STARTED    NO
     4 so3       so3          STARTED    NO

     3 Satz/Sätze ausgewählt.
```

John setzt diesen Befehl ab, um Member 0 zu starten: `db2start member 0`. Er fragt den Status aller Member in der Instanz mit der folgenden Abfrage ab:

```
SELECT ID,
       varchar(HOME_HOST,10) AS HOME_HOST,
       varchar(CURRENT_HOST,10) AS CUR_HOST,
       varchar(STATE,21) AS STATE,
       ALERT
FROM SYSIBMADM.DB2_MEMBER
```

Und er erhält die folgende Ausgabe:

```
ID     HOME_HOST CUR_HOST     STATE      ALERT
------ --------- ------------ ---------- -----
     0 so1       so1          STARTED    NO
```

```
2 so2       so2          STARTED    NO
4 so3       so3          STARTED    NO

3 Satz/Sätze ausgewählt.
```

Stoppen eines Members

DB2-Member werden durch einen global ausgeführten Befehl **db2stop** oder einen individuell ausgeführten Befehl **db2stop** gestoppt. Dieser Abschnitt behandelt in erster Linie das Stoppen eines einzelnen Members.

Informationen zu diesem Vorgang

Setzen Sie den Befehl `db2stop Member` ab, wenn Sie ein Member auf einem Host stoppen, jedoch gleichzeitig andere Instanzprozesse auf demselben Host weiterhin ausführen möchten. Dies bedeutet, dass andere Member oder Cluster-Caching-Funktionen auf demselben Host nicht betroffen sind. Beachten Sie jedoch, dass das Stoppen eines Members keine ausreichende Voraussetzung für das Ausführen von Wartungstasks auf dem Host ist, da dieser trotzdem ein funktionsfähiges Failover-Ziel für andere Member bleibt.

Einschränkungen

Sie können ein Member nicht stoppen, wenn aktive Datenbankverbindungen auf dem Member vorhanden sind. Wenn sich das Member im Modus für einen Light-Neustart befindet und die Recovery nach dem Absturz eines Members noch nicht durchgeführt wurde (das heißt, mindestens eine der Datenbanken auf diesem Member ist inkonsistent), kann es nicht mithilfe des Befehls **db2stop** gestoppt werden. Es kann jedoch mit der Option **FORCE** gestoppt werden. Es wird jedoch ausdrücklich empfohlen, ein Member, für das eine Recovery nach einem Absturz ausgeführt wird, nicht zu stoppen.

Vorgehensweise

Setzen Sie zum Stoppen eines Members den folgenden Befehl ab:

`db2stop member` *`member-id`*

Datenbankaktivierung mit DB2 pureScale Feature

Eine Datenbank in einer DB2 pureScale-Umgebung kann explizit mit dem Befehl **ACTIVATE DATABASE** oder implizit beim erfolgreichen Verbindungsaufbau des ersten Clients aktiviert werden.

Vorbereitende Schritte

Stellen Sie sicher, dass Sie über eine der folgenden Berechtigungsstufen verfügen:

- SYSMAINT
- SYSCTRL
- SYSADM

Informationen zu diesem Vorgang

In einer DB2 pureScale-Umgebung bleibt die Datenbank auch dann aktiv, wenn der letzte Benutzer die Verbindung getrennt hat. Um die Datenbank herunterzufahren, müssen Sie einen Befehl **DEACTIVATE DATABASE** oder den Befehl **db2stop** absetzen.

Vorgehensweise

Um eine Datenbank in einer DB2 pureScale-Umgebung zu aktivieren, setzen Sie den Befehl **ACTIVATE DATABASE** ab, mit dem die Datenbank automatisch für alle Member aktiviert wird.

Beispiel

Um die Datenbank TEST für alle Member zu aktivieren, führen Sie den folgenden Befehl aus:

```
DB2 ACTIVATE DATABASE TEST
```

Datenbankinaktivierung mit DB2 pureScale Feature

In einer DB2 pureScale-Umgebung kann eine Datenbank, die explizit über den Befehl **ACTIVATE DATABASE** oder implizit über eine Benutzerverbindung aktiviert wurde, nur über einen Befehl **DEACTIVATE DATABASE** inaktiviert werden.

Vorbereitende Schritte

Stellen Sie sicher, dass Sie über eine der folgenden Berechtigungsstufen verfügen:

- SYSMAINT
- SYSCTRL
- SYSADM

Informationen zu diesem Vorgang

In einer DB2 pureScale-Umgebung bleibt die Datenbank auch dann aktiv, wenn der letzte Benutzer die Verbindung getrennt hat. Um die Datenbank herunterzufahren, müssen Sie einen Befehl **DEACTIVATE DATABASE** oder den Befehl **db2stop** absetzen.

Vorgehensweise

1. Um eine Datenbank in einer DB2 pureScale-Umgebung zu inaktivieren, setzen Sie den Befehl **DEACTIVATE DATABASE** ab, mit dem die Datenbank automatisch für alle Member inaktiviert wird.
2. Zur Inaktivierung eines bestimmten Members führen Sie den Befehl **DEACTIVATE DATABASE** mit dem Parameter **MEMBER** aus.

Beispiel

Beispiel 1

Setzen Sie zum Inaktivieren der Datenbank TEST den folgenden Befehl ab:

```
DB2 DEACTIVATE DATABASE TEST
```

Beispiel 2

Setzen Sie zum Inaktivieren eines bestimmten Members den folgenden Befehl ab:

```
DB2 DEACTIVATE DATABASE TEST MEMBER 10
```

Wartung in einer DB2 pureScale-Umgebung

Einer der Vorteile von IBM DB2 pureScale Feature besteht darin, dass eine kontinuierliche Verfügbarkeit der Datenbank während geplanter Wartungsaktivitäten sichergestellt wird.

Mit DB2-Cluster-Services können Sie für die Cluster-Caching-Funktionen in Ihrer DB2 pureScale-Instanz eine schrittweise Wartung durchführen. In ähnlicher Weise können Member oder Memberhosts, die gewartet werden sollen, ohne großen Aufwand aus der DB2 pureScale-Instanz entfernt und nach der Wartung wieder integriert werden.

Ab IBM Data Studio Version 3.1 kann der Taskassistent für Folgendes verwendet werden: Versetzen von Zielhosts in den Wartungsmodus bzw. Entfernen aus dem Wartungsmodus. Taskassistenten führen durch den Prozess der Definition von Optionen, der Prüfung automatisch generierter Befehle für die jeweilige Task und der Ausführung dieser Befehle. Weitere Einzelheiten finden Sie in Verwalten von Datenbanken mit Taskassistenten.

Ausführen von Wartungstasks auf einem Member-Host

Sie können Wartungstasks auf einem Member-Host durchführen bzw. Aktualisierungen auf diesen Host anwenden, ohne dabei die Verfügbarkeit der Datenbanken in der DB2 pureScale-Instanz zu beeinträchtigen.

Vorbereitende Schritte

Alle DB2-Serverprozesse und alle anderen Prozesse, die auf die von DB2-Cluster-Services verwalteten Dateisysteme zugreifen, müssen vor dem Ausführen dieser Task beendet werden. Alle Instanzen müssen auf dem Host gestoppt werden, bevor der Host in den Wartungsmodus versetzt werden kann. Wenn die Instanz auf dem Host gestoppt ist, ist der Host kein funktionsfähiges Recoveryziel für ausgefallene Member mehr.

Zur Ausführung dieser Task müssen Sie Administrator der DB2-Cluster-Services sein.

Vorgehensweise

1. Führen Sie als Instanzbenutzer die folgenden Schritte aus:
   a. Führen Sie eine QUIESCE-Operation für das Member mithilfe des Parameters **QUIESCE** des Befehls **db2stop** wie folgt aus:
      ```
      db2stop member member-id quiesce 30
      ```
      Dabei ist *member-id* das Member, das Sie in den Wartungsmodus versetzen wollen.
      Weitere Informationen zum Versetzen eines Members in den Quiescemodus finden Sie in „Versetzen eines Members in den Quiescemodus" auf Seite 96.
   b. Bevor Sie den Host stoppen, stellen Sie sicher, dass sich dort nur die Member befinden, die auf dem Host gewartet werden sollen. Stoppen Sie die Instanz auf dem Member-Host wie folgt:
      ```
      db2stop instance on hostname
      ```
      Dabei ist *hostname* der Name des Hosts des betreffenden Members oder der betreffenden CF.
2. Führen Sie als DB2-Cluster-Services-Administrator die folgenden Schritte aus:
   a. Versetzen Sie den Cluster-Manager auf dem Host in den Wartungsmodus, indem Sie den folgenden Befehl absetzen:
      ```
      DB2DIR/bin/db2cluster -cm -enter -maintenance
      ```
      Dabei steht *DB2DIR* für die Installationsposition Ihrer DB2-Kopie.

b. Versetzen Sie den Clusterdateisystemservice auf dem Host in den Wartungsmodus, indem Sie folgenden Befehl eingeben:
```
DB2DIR/bin/db2cluster -cfs -enter -maintenance
```
Dabei steht *DB2DIR* für die Installationsposition Ihrer DB2-Kopie.
c. Führen Sie einen Warmstart des Hosts durch.

Nächste Schritte

1. Führen Sie als DB2-Cluster-Services-Administrator die folgenden Schritte aus:
   a. Führen Sie alle geplanten Wartungsaktivitäten durch. Nehmen Sie z. B. Änderungen an der Hardwarekonfiguration vor. Beachten Sie, dass es für Aktualisierungen an den DB2 pureScale-Softwarekomponenten erforderlich ist, dass zuvor alle Hosts in den Wartungsmodus versetzt werden.
   b. Beenden Sie den Cluster-Manager-Wartungsmodus auf dem Host, indem Sie den folgenden Befehl absetzen:
   ```
   DB2DIR/bin/db2cluster -cm -exit -maintenance
   ```
   Dabei steht *DB2DIR* für die Installationsposition Ihrer DB2-Kopie.
   c. Beenden Sie den Clusterdateisystem-Wartungsmodus auf dem Host, indem Sie den folgenden Befehl absetzen:
   ```
   DB2DIR/bin/db2cluster -cfs -exit -maintenance
   ```
   Dabei steht *DB2DIR* für die Installationsposition Ihrer DB2-Kopie.
2. Führen Sie als Instanzbenutzer die folgenden Schritte aus:
   a. Starten Sie die Instanz auf dem Host erneut, indem Sie den folgenden Befehl absetzen:
   ```
   db2start instance on hostname
   ```
   Dabei ist *hostname* der Name des Hosts des betreffenden Members oder der betreffenden CF.
   b. Starten Sie das Member erneut, indem Sie den folgenden Befehl absetzen:
   ```
   db2start member member-id
   ```
   Dabei ist *member-id* das Member, das Sie in den Wartungsmodus versetzen wollen.

Ersetzen beider Cluster-Caching-Funktionen

Sie können beide Cluster-Caching-Funktionen mithilfe eines schrittweisen Upgradeverfahrens ersetzen, mit dem sichergestellt wird, dass es zu keinem Ausfall Ihrer DB2 pureScale-Instanz kommen kann.

Vorbereitende Schritte

Sie müssen über Rootberechtigung verfügen, um diese Task ausführen zu können.

Vergewissern Sie sich, welche Cluster-Caching-Funktion zurzeit die primäre ist, sodass Sie diese nicht versehentlich stoppen. Darüber hinaus muss sich die sekundäre Cluster-Caching-Funktion im Status PEER befinden. Sie können beides erledigen, indem Sie die Verwaltungssicht DB2_CF wie folgt abfragen: `SELECT * FROM SYSIBMADM.DB2_CF`

Vorgehensweise

Führen Sie die folgenden Schritte aus, um beide Cluster-Caching-Funktionen (CFs) zu ersetzen:

1. Stoppen Sie die sekundäre Cluster-Caching-Funktion mithilfe des folgenden Befehls: `db2stop CF CF-ID`. Weitere Informationen zum Stoppen einer Cluster-Caching-Funktion, finden Sie im Abschnitt „Stoppen einer Cluster-Caching-Funktion" auf Seite 88.
2. Stoppen Sie die Instanz der sekundären Cluster-Caching-Funktion mithilfe des folgenden Befehls, um sicherzustellen, dass keine weiteren DB2-Prozesse ausgeführt werden: `db2stop instance on hostname`.
3. Löschen Sie die Cluster-Caching-Funktion mithilfe des folgenden Befehls: `db2iupdt -drop -cf hostname instanzname`.
4. Fügen Sie mithilfe des folgenden Befehls den neuen Host hinzu: `db2iupdt -add -cf hostname:netzname instanzname`.
5. Ermitteln Sie die ID für die neu hinzugefügte Cluster-Caching-Funktion mithilfe der folgenden Anweisung: `SELECT * from SYSIBMADM.DB2_CF`.
6. Starten Sie die neu hinzugefügte Cluster-Caching-Funktion mithilfe des folgenden Befehls: `db2start CF CF-ID`. Weitere Informationen zum Starten einer Cluster-Caching-Funktion finden Sie im Abschnitt „Starten einer Cluster-Caching-Funktion" auf Seite 86.
7. Sobald die neue Cluster-Caching-Funktion (jetzt die sekundäre Cluster-Caching-Funktion) den Peerzustand erreicht, stoppen Sie die primäre Cluster-Caching-Funktion (wie in Schritt 1 und 2 beschrieben). Die DB2-Cluster-Services führen eine Übertragung der primären Rolle auf die sekundäre Cluster-Caching-Funktion aus (Failover).
8. Wiederholen Sie die Schritte 3 bis 6, um die alte primäre Cluster-Caching-Funktion zu ersetzen und sie als sekundäre Cluster-Caching-Funktion erneut zu starten.

Hinzufügen einer Platte zum gemeinsam genutzten Dateisystem

Nach dem Erstellen der gemeinsam genutzten Dateisysteme kann es erforderlich sein, zusätzliche Platten zum Dateisystem hinzuzufügen.

Vorbereitende Schritte

Die Benutzer-ID, die diesen Befehl ausführt, muss Eigner der Platten sein und Lese- und Schreibzugriff auf sie haben. Die Platten dürfen zurzeit nicht von einem anderen Dateisystem auf dem lokalen Host verwendet werden. Um diese Task durchzuführen, müssen Sie entweder die Benutzer-ID sein, die das Dateisystem erstellt hat, oder der Administrator der DB2-Cluster-Services.

Vorgehensweise

Sobald die Platte für die Hosts physisch verfügbar ist, können Sie sie mit dem Befehl **db2cluster** dem Dateisystem hinzufügen, das mehr Speicherplatz benötigt. Beispiel:

```
db2cluster -add -filesystem dateisystemname -disk plattenname
```

Ergebnisse

Wenn diese Task abgeschlossen ist, wird die Platte zum Dateisystem hinzugefügt, und das Stripe-Set für das Dateisystem enthält die entsprechende Platte.

Nächste Schritte

Wenn Sie sicherstellen möchten, dass das Dateisystem über alle Platten verteilt ist, müssen Sie einen Neuausgleich durchführen.

Entfernen einer Platte aus dem gemeinsam genutzten Dateisystem:

Wenn Sie feststellen, dass Ihr System nicht alle auf ihm befindlichen Platten benötigt oder dass eine bestimmte Platte besser in einem Dateisystem mit mehr Speicher implementiert werden kann, können Sie die gewünschte Platte ohne großen Aufwand mit dem Befehl **db2cluster** entfernen.

Vorbereitende Schritte

Stellen Sie sicher, dass auf den verbleibenden Platten im Dateisystem ausreichend Speicherplatz vorhanden ist, um die Dateien, die sich auf der Platte befinden, die entfernt werden soll, unterzubringen.

Informationen zu diesem Vorgang

Durch diese Task wird eine Platte aus dem Dateisystem entfernt und für andere Zwecke verfügbar gemacht.

Einschränkungen

- Der Instanzeigner kann diese Task nur ausführen, wenn er das Dateisystem erstellt hat. Ansonsten kann nur der Administrator der DB2-Cluster-Services eine Platte aus einem Dateisystem entfernen.
- Wenn nur eine Platte im Dateisystem übrig ist, können Sie diese Platte mit dieser Methode nicht entfernen. Löschen Sie stattdessen das Dateisystem, wie im Abschnitt com.ibm.db2.luw.admin.sd.doc/doc/t0056124.dita beschrieben.
- Sie können den Plattentiebreaker nicht entfernen. Informationen zur Ermittlung, ob eine Platte ein Tiebreaker ist, finden Sie in com.ibm.db2.luw.admin.sd.doc/doc/c0056704.dita.
- Sie können jeweils nur eine Platte gleichzeitig aus einem Dateisystem entfernen.

Vorgehensweise

Verwenden Sie zum Entfernen einer Platte aus dem gemeinsam genutzten Dateisystem den Befehl **db2cluster**:

```
db2cluster -cfs -remove -filesystem dateisystemname -disk plattenname
```

Nächste Schritte

Sobald die Platte aus dem Dateisystem entfernt ist, führen Sie einen Neuausgleich durch, um sicherzustellen, dass das Dateisystem über alle Platten verteilt ist. Informationen zu diesem Vorgang finden Sie im Abschnitt „Neuausgleichen eines Dateisystems".

Neuausgleichen eines Dateisystems:

Sie können Ihr Dateisystem nach dem Hinzufügen oder Entfernen von Platten neu ausgleichen. Mithilfe des Befehls **db2cluster** können Sie einen solchen Neuausgleich durchführen.

Vorbereitende Schritte

Stellen Sie sicher, dass das Dateisystem angehängt (Mount) ist, bevor Sie diesen Befehl ausführen. Da bei einem Neuausgleich eine intensive Dateieingabe und -ausgabe erfolgt, wird empfohlen, den Neuausgleich bei einer reduzierten Systemaktivität im Cluster durchzuführen.

Informationen zu diesem Vorgang

Durch diese Task wird das Dateisystem in der Weise ausgeglichen, dass die auf Platten gespeicherten Daten durch ein erneutes Striping auf alle Platten im Dateisystem verteilt werden. Der Instanzeigner kann diese Task nur ausführen, wenn er das Dateisystem erstellt hat. Ansonsten kann nur der Administrator der DB2-Cluster-Services ein Dateisystem neu ausgleichen.

Vorgehensweise

Verwenden Sie den Befehl **db2cluster** wie folgt, um das Dateisystem neu auszugleichen:

```
db2cluster -rebalance -filesystem dateisystemname
```

Versetzen eines Members in den Quiescemodus

Gewisse Umstände (beispielsweise Wartungsoperationen) können es erfordern, dass Sie ein Member zeitweilig aus dem Cluster entfernen.

Informationen zu diesem Vorgang

In einer DB2 pureScale-Umgebung verfügen die Befehle **db2stop** und **STOP DATABASE MANAGER** über den optionalen Parameter **QUIESCE**. Mit diesem Parameter können Sie alle Aktivitäten eines einzelnen Members auslaufen lassen und diesen beenden. Während die Aktivitäten des Members auslaufen, leiten die automatische Clientweiterleitung (ACR, Automatic Client Reroute) und die Lastausgleichfunktion neue Transaktionen und neue Verbindungsanforderungen an andere Member um.

Wenn ein Befehl **db2stop** mit dem optionalen Parameter **QUIESCE** für ein Member abgesetzt wird, unterliegen Anwendungen den folgenden Bedingungen:

- Wenn Sie ein Zeitlimit angeben, können Anwendungen innerhalb dieses Limits ihre Arbeitseinheiten beenden.
- Wird kein Zeitlimit (oder der Wert -1) angegeben, wartet der Server so lange, bis alle aktiven Transaktionen und zugehörigen Verbindungen für das Member beendet wurden.
- Wenn Sie für den Zeitlimitwert den Wert 0 (null) angeben, werden Verbindungen unverzüglich zwangsweise getrennt.

Anmerkung: Wenn ein Befehl zum Versetzen eines Members in den Quiescemodus (ohne Zeitlimit oder mit einem Zeitlimitwert ungleich null) ausgeführt wird, werden Transaktionen für die verzögerte Verbindungsbeendigung markiert. Wenn eine Transaktion erfolgreich eine Commit- oder Rollback-Operation durchführt, wird die Verbindung beendet, es sei denn, eine der folgenden Bedingungen trifft zu:

- Die Verbindung verwendet globale Variablen.
- Es wird ein verschlüsseltes Kennwort verwendet.
- Es gibt einen offenen WITH HOLD-Cursor.
- Es werden deklarierte temporäre Tabellen (DGTT) verwendet.

- Eine Umsetzungsgruppe (TRANSFORM GROUP) ist festgelegt.
- Die Berechtigungs-ID der Sitzung (SESSION AUTHID) wurde geändert.
- Es werden PL/SQL-Pakete oder SQL/PL-Module verwendet.
- Es werden Cursorvariablen verwendet.
- Es werden Sequenzwerte verwendet.
- Es werden erstellte temporäre Tabellen (CGTT) mit der Klausel PRESERVE ROWS verwendet.
- In einem Paket vorbereitetes dynamisches SQL wurde mit KEEPDYNAMIC YES gebunden. Diese Einschränkung gilt nicht, wenn Anweisungen in einer gespeicherten Prozedur bzw. einer benutzerdefinierten Funktion oder über nicht integrierte IBM APIs wie CLI/JDBC/ODBC/.NET vorbereitet werden.

Wenn eine der aufgelisteten Bedingungen zutrifft, müssen Sie diese Bedingung entweder entfernen (wie WITH HOLD-Cursor) oder die Verbindung explizit beenden. Wenn keine dieser Bedingungen zutrifft, wird die Clientverbindung am nächsten Transaktionsendpunkt beendet. Wenn Ihr Client (.NET, CLI, JDBC) eine nahtlose automatische Clientweiterleitung (ACR) unterstützt, werden Ihre Verbindungen automatisch an ein anderes Member weitergeleitet.

Vorgehensweise

Führen Sie den Befehl **db2stop** oder den Befehl **STOP DATABASE MANAGER** aus und geben Sie dabei den Parameter **QUIESCE** an.
In diesem Beispiel wird Member 2 in den Offline-Modus versetzt, und die aktive Workload muss in 30 Minuten beendet werden. Nach 30 Minuten werden alle Anwendungen zwangsweise vom Member getrennt.

```
db2stop MEMBER 2 QUIESCE 30
```

Um Member 2 wieder online zu stellen, setzen Sie für das Member den Befehl **db2start** ab.

```
db2start MEMBER 2
```

Beispiel

Dieses Beispiel verwendet die API db2InstanceStop, um Member 10 in den Offline-Modus zu versetzen.

```
struct sqlca sqlca;     // SQL-Kommunikationsbereich zur Aufnahme des SQLCODE
struct db2InstanceStopStruct instanceStopStruct;
struct db2StopOptionsStruct  stopOptions;

instanceStopStruct.iIsRemote = FALSE; // Lokale Demoinstanz
instanceStopStruct.piRemoteInstName = NULL;
instanceStopStruct.piCommData = NULL; // DAS nicht berücksichtigen
instanceStopStruct.piStopOpts = &stopOptions;

stopOptions.iOption = SQLE_QUIESCE; // QUIESCE-Option für Member
stopOptions.iIsType = TRUE;
stopOptions.iType = DB2_NODE_MEMBER;
stopOptions.iIsNodeNum = TRUE;
stopOptions.iNodeNum = 10;
stopOptions.iQuiesceDeferMinutes = 0; // Kein explizites Zeitlimit

// Schließlich Aufrufen der API, um Instanz herunterzufahren
db2InstanceStop(db2Version1010, &instanceStopStruct, &sqlca);
```

Dieses Beispiel verwendet die API db2InstanceStop, um Member 10 in den Offline-Modus zu versetzen und ein Zeitlimit von 5 Minuten anzugeben.

```
struct sqlca sqlca;     // SQL-Kommunikationsbereich zur Aufnahme des SQLCODE
struct db2InstanceStopStruct instanceStopStruct;
struct db2StopOptionsStruct  stopOptions;

instanceStopStruct.iIsRemote = FALSE; // Lokale Demoinstanz
instanceStopStruct.piRemoteInstName = NULL;
instanceStopStruct.piCommData = NULL; // DAS nicht berücksichtigen
instanceStopStruct.piStopOpts = &stopOptions;

stopOptions.iOption = SQLE_QUIESCE; // QUIESCE-Option für Member
stopOptions.iIsType = TRUE;
stopOptions.iType = DB2_NODE_MEMBER;
stopOptions.iIsNodeNum = TRUE;
stopOptions.iNodeNum = 10;
stopOptions.iQuiesceDeferMinutes = 5; // Zeitlimit von fünf Minuten

// Schließlich Aufrufen der API, um Instanz herunterzufahren
db2InstanceStop(db2Version1010, &instanceStopStruct, &sqlca);
```

Versetzen von Hosts in den Wartungsmodus

Wenn Sie Software-Updates auf DB2-Cluster-Services anwenden, müssen Sie den Zielhost in den Wartungsmodus versetzen. Sie können einen Host auch in den Wartungsmodus versetzen, wenn Sie sicherstellen wollen, dass Member oder Cluster-Caching-Funktionen auf dem Host nicht erneut gestartet werden, während Sie Updates am Betriebssystem oder an der Hardware des Hosts vornehmen.

Vorbereitende Schritte

Alle DB2-Serverprozesse und alle anderen Prozesse, die auf die von DB2-Cluster-Services verwalteten Dateisysteme zugreifen, müssen vor dem Ausführen des Befehls beendet werden. Alle Instanzen müssen auf dem Host gestoppt werden, bevor der Host in den Wartungsmodus versetzt werden kann. Wenn die Instanz auf dem Host gestoppt ist, ist der Host kein funktionsfähiges Recoveryziel für ausgefallene Member mehr.

Zur Ausführung dieser Task müssen Sie Administrator der DB2-Cluster-Services sein.

Ein Host kann nicht in den Wartungsmodus versetzt werden, wenn noch irgendwelche Serverprozesse auf dem Host aktiv sind. Selbst wenn Sie die Option **-force** im Befehl angeben, wird dadurch nur der Cluster-Manager und das gemeinsam genutzte Dateisystem beendet.

Um einen Ausfall Ihrer DB2 pureScale-Instanz zu verhindern, halten Sie sich an folgende Empfehlungen:

- Versetzen Sie die Hosts der beiden Cluster-Caching-Funktionen nicht gleichzeitig in den Wartungsmodus.
- Versetzen Sie nicht alle Hosts mit Membern gleichzeitig in den Wartungsmodus.
- Versetzen nicht mehr als 50 % der Hosts in der DB2 pureScale-Instanz gleichzeitig in den Wartungsmodus.

Informationen zu diesem Vorgang

Je mehr Hosts sich im Wartungsmodus befinden, desto weniger Hosts stehen im Falle eines Ausfalls oder einer Störung für eine Recovery zur Verfügung. Aus diesem Grund wird empfohlen, nicht mehr als einen Host gleichzeitig in den Wartungsmodus zu versetzen.

Vorgehensweise

- Wenn Sie einen Host in den Wartungsmodus versetzen wollen, beachten Sie die Informationen in „Ausführen von Wartungstasks auf einem Member-Host" auf Seite 92.
- Um den Wartungsmodus für alle Hosts durchzusetzen, beachten Sie die Informationen in „Versetzen eines Clusters in den Wartungsmodus".

Zugehörige Informationen:

Versetzen eines Clusters in den Wartungsmodus

Sie können ein Cluster in den Wartungsmodus versetzen, wenn Sie Updates am Betriebssystem oder an der Hardware für die Hosts im Cluster vornehmen.

Zur Ausführung dieser Task müssen Sie Administrator der DB2-Cluster-Services sein.

Vorgehensweise

1. Führen Sie als Instanzbenutzer die folgenden Schritte aus:
   a. Stoppen Sie den Datenbankmanager auf allen Hosts, indem Sie den folgenden Befehl auf einem der Hosts absetzen:
      ```
      su -iname
      db2stop
      exit
      ```
      Dabei ist *iname* der Name des Instanzeigners.
   b. Stoppen Sie auf jedem Host die DB2-Instanz, indem Sie den folgenden Befehl absetzen:
      ```
      db2stop instance on hostname
      ```
      Dabei ist *hostname* der Hostname für ein bestimmtes Member oder eine bestimmte CF und der Befehl `db2stop instance on hostname` wird für jeden Host im Cluster ausgeführt.
2. Führen Sie als DB2-Cluster-Services-Administrator die folgenden Schritte aus:
   a. Setzen Sie den folgenden Befehl für jede Ressourcengruppe ab, um die Ressourcengruppen abzuhängen:
      ```
      rgreq -o stop ressourcengruppenname
      ```
      Dabei ist *ressourcengruppenname* der Name jeder Ressourcengruppe, die bei Ausführung des Befehls `lsrg | grep db2mnt` zurückgegeben wird. Sie können mithilfe des Befehls `lssam -g ressourcengruppenname` prüfen, ob die Ressourcengruppe offline ist.
   b. Versetzen Sie den Cluster-Manager auf allen Hosts in den Wartungsmodus, indem Sie den folgenden Befehl absetzen:
      ```
      DB2DIR/bin/db2cluster -cm -enter -maintenance -all
      ```
      Dabei steht *DB2DIR* für die Installationsposition Ihrer DB2-Kopie.
   c. Versetzen Sie den Clusterdateisystemservice auf den Hosts in den Wartungsmodus, indem Sie folgenden Befehl eingeben:
      ```
      DB2DIR/bin/db2cluster -cfs -enter -maintenance -all
      ```
      Dabei steht *DB2DIR* für die Installationsposition Ihrer DB2-Kopie.
   d. Prüfen Sie den Wert des Konfigurationsparameters **autoload**, indem Sie den folgenden Befehl **mmlsconfig** absetzen:
      ```
      mmlsconfig | grep autoload
      ```

Wenn der Wert des Konfigurationsparameters 'autoload' auf YES eingestellt ist, setzen Sie den folgenden Befehl ab, um den Wert in NO zu ändern, sodass die Knoten nicht automatisch gestartet werden:

```
mmchconfig autoload=no -N all
```

e. Stoppen Sie alle Hosts.

Nächste Schritte

1. Führen Sie als DB2-Cluster-Services-Administrator die folgenden Schritte aus:
   a. Führen Sie alle geplanten Wartungsaktivitäten durch. Installieren Sie beispielsweise ein Fixpack-Update oder nehmen Sie Änderungen an der Netztopologie Ihrer DB2 pureScale-Umgebung vor.
   b. Prüfen Sie den Wert des Konfigurationsparameters **autoload**, indem Sie den folgenden Befehl **mmlsconfig** absetzen:

      ```
      mmlsconfig | grep autoload
      ```

      Wenn der Wert des Konfigurationsparameters **autoload** auf NO eingestellt ist, setzen Sie den folgenden Befehl ab, um den Wert in YES zu ändern:

      ```
      mmchconfig autoload=yes -N all
      ```

   c. Stellen Sie sicher, dass alle Member und die Domäne aktiv sind, indem Sie den folgenden Befehl absetzen:

      ```
      DB2DIR/bin/db2cluster -cm -list -host -state
      ```

      Dabei steht *DB2DIR* für die Installationsposition Ihrer DB2-Kopie.
   d. Beenden Sie den Cluster-Manager-Wartungsmodus auf dem Host, indem Sie den folgenden Befehl absetzen:

      ```
      DB2DIR/bin/db2cluster -cm -exit -maintenance
      ```

      Dabei steht *DB2DIR* für die Installationsposition Ihrer DB2-Kopie.
   e. Setzen Sie den Status der Ressourcengruppe zurück, indem Sie den folgenden Befehl für jede Ressourcengruppe absetzen:

      ```
      rgreq -o cancel ressourcengruppenname
      ```

      Dabei ist *ressourcengruppenname* der Name jeder Ressourcengruppe, die bei Ausführung des Befehls `lsrg | grep db2mnt` zurückgegeben wird.
   f. Beenden Sie den Clusterdateisystem-Wartungsmodus auf den Hosts, indem Sie den folgenden Befehl absetzen:

      ```
      DB2DIR/bin/db2cluster -cfs -exit -maintenance -all
      ```

      Dabei steht *DB2DIR* für die Installationsposition Ihrer DB2-Kopie. Im Falle einer Zeitlimitüberschreitung prüfen Sie den Status des Clusterdateisystems, indem Sie den Befehl `DB2DIR/bin/db2cluster -cfs -list -host -state` absetzen.
2. Führen Sie als Instanzbenutzer die folgenden Schritte aus:
   a. Starten Sie auf jedem Host die DB2-Instanz, indem Sie den folgenden Befehl absetzen:

      ```
      db2start instance on hostname
      ```

      Dabei ist *hostname* der Hostname für ein bestimmtes Member oder eine bestimmte CF und der Befehl `db2start instance on hostname` wird für jeden Host im Cluster ausgeführt.
   b. Starten Sie den Datenbankmanager, indem Sie den folgenden Befehl eingeben:

```
su -iname
db2start exit
```

Dabei ist *iname* der Name des Instanzeigners.

Versetzen eines DB2-Members oder einer Cluster-Caching-Funktion

Es gibt eine Reihe von Gründen zum Versetzen eines DB2-Members oder einer Cluster-Caching-Funktion von einem Host zu einem anderen. Diese Task zeigt einige mögliche Szenarios sowie einige zusätzlich zu beachtende Faktoren auf.

Informationen zu diesem Vorgang

Wenn mit der Operation **db2iupdt -add** oder **-drop** eine Änderung in einer DB2 pureScale-Umgebung vorgenommen wird, sollte für die Datenbank möglicherweise ein Backup durchgeführt werden. Dieses Backup muss von einem der vorher vorhandenen Instanzmember aus durchgeführt werden.

Einschränkungen

Es werden maximal zwei Cluster-Caching-Funktionen unterstützt. Wenn Sie eine davon versetzen möchten, müssen Sie die andere zunächst löschen und dann die zweite zu einem neuen Host hinzufügen. Wenn Sie versuchen, eine dritte Cluster-Caching-Funktion hinzuzufügen, ohne zuvor eine der beiden vorhandenen zu löschen, empfangen Sie einen Fehler.

Es müssen immer mindestens ein DB2-Member und eine Cluster-Caching-Funktion vorhanden sein. Wenn Sie eines bzw. eine davon versetzten möchten, müssen Sie zunächst das neue DB2-Member bzw. die Cluster-Caching-Funktion auf einem neuen Host hinzufügen, bevor die entsprechende Vorlage gelöscht werden kann. Wenn Sie versuchen, zuerst das letzte verbleibende Member oder die letzte verbleibende Cluster-Caching-Funktion zu löschen, empfangen Sie einen Fehler.

Vorgehensweise

- Gehen Sie wie folgt vor, um das einzige DB2-Member zu versetzen:
  1. Stoppen Sie die DB2 pureScale-Instanz auf allen Hosts mithilfe des Befehls **db2stop**. Dieser Schritt ist erforderlich, da die Befehle **db2iupdt -add** und **-drop** offline ausgeführt werden müssen.
  2. Fügen Sie ein neues Member hinzu. Durch diese Hinzufügung wird sichergestellt, dass mindestens ein DB2-Member vorhanden ist. Fügen Sie beispielsweise ein DB2-Member mit dem Namen `Member2` und dem Netznamen `Netname2` zu der DB2 pureScale-Instanz `sdinstA` hinzu:
     ```
     db2iupdt -add -m Member2:Netname2 sdinstA
     ```
  3. Löschen Sie das ursprüngliche DB2-Member. Löschen Sie beispielsweise das Member mit dem Namen `Member1` und dem Netznamen `Netname1` aus der DB2 pureScale-Instanz `sdinstA`:
     ```
     db2iupdt -drop -m Member1:Netname1 sdinstA
     ```
  4. Führen Sie ein Backup der Datenbank (falls diese wiederherstellbar ist) für diese DB2 pureScale-Instanz von einem Member aus durch, das vor dem Starten dieses Prozesses vorhanden war. Bei wiederherstellbaren Datenbanken ist der Datenbankkonfigurationsparameter **logarchmeth1** oder **logarchmeth2** auf einen anderen Wert als `OFF` eingestellt.
- Gehen Sie wie folgt vor, um eines von mehreren DB2-Membern zu versetzen:

1. Der Befehl **db2iupdt** -add oder -drop muss offline ausgeführt werden. Stoppen Sie daher mithilfe des Befehls **db2stop** die Instanz auf allen Hosts.
2. Löschen Sie die DB2-Member, die Sie versetzen wollen. Löschen Sie beispielsweise das Member mit dem Namen Member2 und dem Netznamen Netname2 aus der DB2 pureScale-Instanz sdinstA:

   ```
   db2iupdt -drop -m Member2:Netname2 sdinstA
   ```

3. Fügen Sie ein neues DB2-Member hinzu. Fügen Sie beispielsweise Member3 mit dem Netznamen Netname3 als Member zu der DB2 pureScale-Instanz sdinstA hinzu:

   ```
   db2iupdt -add -m Member3:Netname3 sdinstA
   ```

4. Führen Sie ein Backup der Datenbank (falls diese wiederherstellbar ist) für diese DB2 pureScale-Instanz von einem Member aus durch, das vor dem Starten dieses Prozesses vorhanden war. Bei wiederherstellbaren Datenbanken ist der Datenbankkonfigurationsparameter **logarchmeth1** oder **logarchmeth2** auf einen anderen Wert als OFF eingestellt.

- Gehen Sie wie folgt vor, um die einzige Cluster-Caching-Funktion zu versetzen:
  1. Der Befehl **db2iupdt** -add oder -drop muss offline ausgeführt werden. Stoppen Sie daher mithilfe des Befehls **db2stop** die Instanz auf allen Hosts.
  2. Fügen Sie eine neue Cluster-Caching-Funktion hinzu. Durch diese Hinzufügung wird sichergestellt, dass mindestens eine Cluster-Caching-Funktion vorhanden ist. Fügen Sie beispielsweise eine Cluster-Caching-Funktion mit dem Namen cf2 und dem Netznamen NetCF2 zu der DB2 pureScale-Instanz sdinstA hinzu:

     ```
     db2iupdt -add -cf cf2:NetCF2 sdinstA
     ```

  3. Löschen Sie die ursprüngliche Cluster-Caching-Funktion. Löschen Sie beispielsweise das Cluster-Caching-Funktion mit dem Namen cf1 und dem Netznamen NetCF1 aus der DB2 pureScale-Instanz sdinstA:

     ```
     db2iupdt -drop -cf cf1:NetCF1 sdinstA
     ```

- Gehen Sie wie folgt vor, um eine der beiden Cluster-Caching-Funktionen zu versetzen:
  1. Der Befehl **db2iupdt** -add oder -drop muss offline ausgeführt werden. Stoppen Sie daher mithilfe des Befehls **db2stop** die Instanz auf allen Hosts.
  2. Es werden maximal zwei Cluster-Caching-Funktionen unterstützt; Sie müssen also zunächst eine löschen. Löschen Sie beispielsweise eine Cluster-Caching-Funktion mit dem Namen cf1 und dem Netznamen NetCF1 aus der DB2 pureScale-Instanz sdinstA:

     ```
     db2iupdt -drop -cf cf1:NetCF1 sdinstA
     ```

  3. Fügen Sie die zweite Cluster-Caching-Funktion auf einem anderen Host hinzu. Fügen Sie beispielsweise eine Cluster-Caching-Funktion mit dem Namen cf2 und dem Netznamen NetCF2 zu der DB2 pureScale-Instanz sdinstA hinzu:

     ```
     db2iupdt -add -cf cf2:NetCF2 sdinstA
     ```

Recovery nach einer fehlgeschlagenen Operation 'db2iupdt -add' oder '-drop' in einer DB2 pureScale-Umgebung

Es gibt eine Reihe von Gründen, warum Operationen des Typs **db2iupdt** -add oder -drop fehlschlagen und die Topologie von DB2 pureScale-Umgebungen in einen inkonsistenten Zustand versetzen können.

Falls für das Problem ein Hardware-Fix erforderlich ist (beispielsweise bei einem Hostausfall), beenden Sie diese Task zuerst. Sobald die Hardware betriebsbereit ist, können Sie mithilfe des Befehls **db2iupdt** mit der Option -fixtopology entweder

einen Rollback für die vorherige Operation **db2iupdt** -add durchführen oder die vorherige Operation **db2iupdt** -drop beenden.

Wenn es sich bei der fehlgeschlagenen Operation um einen Versuch handelte, einen Memberhost mit mehreren DB2-Membern zu löschen, wird mit dem Befehl **db2iupdt** -fixtopology nur die vorherige Operation **db2iupdt**-drop des Members beendet, das fehlgeschlagen ist. Es kann unter Umständen nötig sein, den ursprünglichen Befehl **db2iupdt** -drop erneut auszuführen, um auch die anderen DB2-Member auf dem Memberhost löschen zu können.

Löschen von Ressourcenmodellalerts

In einer DB2 pureScale-Umgebung können hardware- und softwarebezogene Alerts in der Statustabelle für Member, Cluster-Caching-Funktionen (CFs) und für Hosts vorhanden sein. In einigen Fällen sind die Alertbedingungen transient, sodass sich das Alertfeld selbst zurücksetzt. In allen anderen Fällen bleibt das Alertfeld jedoch solange gesetzt, bis der Administrator das Problem löst und das Alertfeld manuell zurücksetzt.

Informationen zu diesem Vorgang

Wenn dieser Befehl ausgeführt wird, werden alle Alerts für die Instanz auf dem angegebenen Member, auf der angegebenen Cluster-Caching-Funktion oder auf dem angegebenen Host gelöscht.

Zur Ausführung dieser Task müssen Sie ein Benutzer der Gruppe SYSADM, SYSMAINT oder SYSADM sein.

Vorgehensweise

Um Alerts für ein Member, eine Cluster-Caching-Funktion oder einen Host zurückzusetzen, setzen Sie den folgenden Befehl ab:

```
db2cluster -clear -alert  [-member member-id | -cf cf-id | -host hostname]
```

Ergebnisse

Nach der Ausführung des Befehls führt das System möglicherweise eine bestimmte Aktion in Bezug auf die Komponente aus, für die der Alert gelöscht wurde. In einigen Fällen wird das Member auf seinen Benutzerhost zurückübertragen. Wenn jedoch ein Fehler beim Light-Neustart den Alert verursacht hat, wird keine Aktion ausgeführt.

Ändern des Quorumtyps des Clusters

Der Tiebreaker der DB2-Cluster-Services wird zu Anfang über die grafische Benutzerschnittstelle (GUI) der Installation bei der Erstellung der ersten Instanz konfiguriert. Ändern Sie den Quorumtyp des Cluster-Managers nur, wenn mit der angegebenen Platte ein Problem auftritt oder wenn die Konfiguration geändert wird.

Vorbereitende Schritte

- Bevor Sie den Quorumtyp ändern können, muss die DB2 pureScale-Instanz auf allen Hosts gestoppt werden. Der Cluster-Manager muss online bleiben.
- Wenn Sie den Quorumtyp in Plattentiebreaker ändern, können Sie die Platte im Einheitenpfadformat oder in Form einer PVID- bzw. WWN-Nummer angeben. Die Platte, die Sie angeben, kann für keine anderen Zwecke verwendet werden.

- Die Plattenoption für den Cluster-Manager kann in Form eines Einheitenpfads (beispielsweise /dev/hdisk0) oder als PVID- bzw. WWN-Nummer eingegeben werden.
- Zur Ausführung dieser Task müssen Sie Administrator der DB2-Cluster-Services sein.

Vorgehensweise

- Um den Quorumtyp in einen Plattentiebreaker zu ändern, setzen Sie den folgenden Befehl ab:

  ```
  db2cluster -cm -set -tiebreaker -disk plattenname
  ```

- Um den Quorumtyp in eine Mehrheitsknotengruppe zu ändern, setzen Sie den folgenden Befehl ab:

  ```
  db2cluster -cm -set -tiebreaker -majority
  ```

Ändern des Quorumtyps für gemeinsam genutzte Dateisysteme

Der Quorumtyp für gemeinsam genutzte Dateisysteme wird während der Konfiguration von IBM DB2 pureScale Feature automatisch zugewiesen. Da der ausgewählte Quorumtyp auf der Anzahl der Hosts im Cluster basiert, müssen Sie nach dem Hinzufügen oder Entfernen von Hosts für die DB2-Instanz den Quorumtyp ändern.

Vorbereitende Schritte

- Bevor Sie den Quorumtyp ändern können, müssen die DB2-Instanz und der Cluster für gemeinsam genutzte Dateisysteme auf allen Hosts beendet werden. Wenn der Cluster für gemeinsam genutzte Dateisysteme noch aktiv ist, führen Sie den Befehl **db2cluster –cfs –stop** aus.
- Wenn Sie den Quorumtyp in Plattentiebreaker ändern wollen, muss die Plattenangabe im Einheitenpfadformat (Beispiel: /dev/hdisk2) erfolgen und die Platte muss von allen Knoten aus zugänglich sein. Im Gegensatz zu dem Plattentiebreaker für den Cluster-Manager kann diese Platte in einem gemeinsam genutzten Dateisystem oder in einer Platte in einem bereits definierten Dateisystem wiederverwendet werden.
- Zur Ausführung dieser Task müssen Sie Administrator der DB2-Cluster-Services sein.

Informationen zu diesem Vorgang

Es gibt zwei Quorumtypen für GPFS:

- *Plattentiebreaker*: eine gemeinsam genutzte Platte, die festlegt, welche Gruppe von Hosts im GPFS-Cluster über das Quorum für den Betrieb verfügt.
- *Mehrheitsknotengruppe*: die Gruppe von Hosts im GPFS-Cluster, von denen die Mehrheit über das Quorum für den Betrieb verfügt.

Der Quorumtyp wird basierend auf der im Abschnitt Tabelle 7 angegebenen Regel automatisch vom DB2-Installationsprogramm ausgewählt.

Tabelle 7. Die Beziehung zwischen der Anzahl von Hosts in einem GPFS-Cluster und dem Quorumtyp

Gesamtzahl der Hosts im GPFS-Cluster	GPFS-Quorumtyp
Gleich oder kleiner 8	Plattentiebreaker
Mehr als 8	Mehrheitsknotengruppe

Wenn sich die Anzahl der Hosts im GPFS von acht Hosts erhöht oder von neun Hosts verringert, wird eine Fehlernachricht zurückgegeben, die anzeigt, dass der Quorumtyp geändert werden muss.

Vorgehensweise

- Um den Quorumtyp in einen Plattentiebreaker zu ändern, setzen Sie den folgenden Befehl ab:

  ```
  db2cluster -cfs -set -tiebreaker -disk plattenname
  ```

- Um den Quorumtyp in eine Mehrheitsknotengruppe zu ändern, setzen Sie den folgenden Befehl ab:

  ```
  db2cluster -cfs -set -tiebreaker -majority
  ```

Ändern der Zeitspanne zum Erkennen von Hostfehlern

Eine schnelle Fehlerkennung ist für eine DB2 pureScale-Instanz extrem wichtig, denn je schneller ein Hostausfall erkannt wird, desto früher kann die Recovery für diesen Ausfall gestartet werden. Unter diesem Aspekt betrachtet, eignet sich eine aggressive Einstellung für die Zeitspanne zum Erkennen von Hostfehlern jedoch nicht für Umgebungen mit hoher Latenz, sodass Benutzer die Einstellung möglicherweise ändern möchten.

Vorbereitende Schritte

- Bevor Sie die Zeitspanne zum Erkennen von Hostfehlern ändern können, müssen die DB2-Instanz und der Cluster für gemeinsam genutzte Dateisysteme auf allen Hosts beendet werden. Wenn der Cluster für gemeinsam genutzte Dateisysteme noch aktiv ist, führen Sie den Befehl **db2cluster –cfs –stop -all** aus.
- Zur Ausführung dieser Task müssen Sie Administrator der DB2-Cluster-Services sein.

Informationen zu diesem Vorgang

Mithilfe des Befehls **db2cluster** können Sie die Erkennungszeit des Fehlers für Ihre DB2 pureScale-Instanz anpassen. Der Befehl gibt an, wie lange es dauert, einen Hostfehler oder eine Netzpartition zu erkennen. Verwenden Sie zum Ermitteln der aktuellen Einstellung den folgenden Befehl:

```
db2cluster -cm -list -HostFailureDetectionTime
```

Vorgehensweise

Verwenden Sie zum Ändern der Zeitspanne zum Erkennen von Hostfehlern den folgenden Befehl:

```
db2cluster -cm -set -option HostFailureDetectionTime -value wert
```

Ergebnisse

Sobald der Befehl erfolgreich ausgeführt wurde, wird die neue Einstellung auf alle DB2 pureScale-Instanzen in der DB2-Cluster-Services-Domäne angewendet.

Teil 2. Datenbanken

Kapitel 5. Datenbanken

Bei einer DB2-Datenbank handelt es sich um eine *relationale Datenbank*. In einer solchen *Datenbank* werden alle Daten in Tabellen gespeichert, die in einer bestimmten Relation zueinander stehen. Diese Relationen zwischen den Tabellen sind so konzipiert, dass die Daten gemeinsam benutzt werden können und dass die doppelte Datenhaltung auf ein Minimum beschränkt werden kann.

Eine *relationale Datenbank* wird als eine Gruppe von Tabellen betrachtet. Die in dieser Datenbank gespeicherten Daten werden auf der Basis des relationalen Datenmodells bearbeitet. Sie enthält eine Gruppe von Objekten, die zum Speichern, Verwalten und für den Zugriff auf die gespeicherten Daten verwendet werden. Zu diesen Objekten gehören z. B. Tabellen, Sichten, Indizes, Funktionen, Trigger sowie Pakete. Objekte werden entweder vom System (integrierte Objekte) oder von einem Benutzer (benutzerdefinierte Objekte) definiert.

Eine *verteilte relationale Datenbank* besteht aus einer Gruppe von Tabellen und anderen Objekten, die über mehrere, jedoch miteinander verbundene Computersysteme verteilt sind. Jedes Computersystem verfügt hierbei über einen Manager für relationale Datenbanken, mit dem die Tabellen in der jeweiligen Umgebung verwaltet werden. Diese Datenbankmanager kommunizieren und kooperieren miteinander auf eine Weise, die es einem bestimmten Datenbankmanager erlaubt, SQL-Anweisungen auf einem der anderen Computersysteme auszuführen.

Eine *partitionierte relationale Datenbank* ist eine relationale Datenbank, deren Daten über mehrere Datenbankpartitionen verteilt sind und partitionsübergreifend verwaltet werden. Diese Verteilung der Daten über mehrere Datenbankpartitionen ist für die meisten SQL-Anweisungen transparent. Bei bestimmten DDL-Anweisungen (DDL, Data Definition Language, Datendefinitionssprache) werden die Datenbankpartitionsinformationen jedoch berücksichtigt (z. B. **CREATE DATABASE PARTITION GROUP**). Bei DDL handelt es sich um eine Untergruppe von SQL-Anweisungen, die zur Beschreibung der Datenrelationen in einer Datenbank verwendet werden.

Bei einer *föderierten Datenbank* handelt es sich um eine relationale Datenbank, deren Datenbestand in mehreren Datenquellen (z. B. in separaten relationalen Datenbanken) gespeichert ist. Die Daten erscheinen für den Benutzer hierbei so, als wären sie alle in einer einzigen, großen Datenbank enthalten. Sie können mithilfe traditioneller SQL-Abfragen abgerufen werden. Änderungen der Daten können explizit für die gewünschte Datenquelle ausgeführt werden.

Entwerfen von Datenbanken

Beim Entwerfen einer Datenbank modellieren Sie ein reales Business-System, das eine Reihe von Entitäten und deren Merkmale (oder *Attribute*) sowie die Regeln bzw. Beziehungen umfasst, die zwischen diesen Entitäten gelten.

Der erste Schritt besteht in der Beschreibung des Systems, das Sie darstellen möchten. Wenn Sie z. B. eine Datenbank für ein Publikationssystem erstellen möchten, dann sollte das System verschiedene Entitätstypen wie beispielsweise Bücher, Autoren, Herausgeber und Verleger umfassen. Für jede dieser Entitäten gibt es bestimmte Informationen (oder Attribute), die Sie aufzeichnen müssen:

- *Bücher:* Titel, ISBN-Nummer, Erscheinungsdatum und -ort, Verleger, ...
- *Autoren:* Name, Anschrift, Telefon- und Faxnummer, E-Mail-Adresse, ...

- *Herausgeber:* Name, Anschrift, Telefon- und Faxnummer, E-Mail-Adresse, ...
- *Verleger:* Name, Anschrift, Telefon- und Faxnummer, E-Mail-Adresse, ...

Ihre Datenbank muss nicht nur diese Entitätstypen sowie die zugehörigen Attribute darstellen, sondern Sie müssen auch in der Lage sein, diese Entitäten in Beziehung zueinander zu setzen. Sie müssen z. B. die Relation darstellen können, die zwischen Büchern und ihren Autoren, zwischen Büchern/Autoren und Verlegern sowie zwischen Büchern/Autoren und Herausgebern besteht.

Zwischen den Entitäten einer Datenbank können drei verschiedene Typen von Beziehungen bestehen:

Eins-zu-eins-Beziehung
Bei diesem Beziehungstyp ist jeder Instanz einer Entität genau eine Instanz der anderen Entität zugeordnet. Momentan besteht in dem oben beschriebenen Szenario keine Eins-zu-eins-Beziehung.

Eins-zu-viele-Beziehung
Bei diesem Beziehungstyp ist jeder Instanz einer Entität mindestens eine Instanz einer anderen Entität zugeordnet. Ein Autor kann z. B. mehrere Bücher geschrieben haben, für die einzelnen Bücher gibt es jedoch nur einen Autor. Dies ist der Beziehungstyp, der in relationalen Datenbanken am häufigsten modelliert wird.

Viele-zu-viele-Beziehungen
Bei diesem Beziehungstyp besteht eine Beziehung zwischen mehreren Instanzen einer bestimmten Entität zu mindestens einer Instanz einer anderen Entität. So kann es z. B. der Fall sein, dass mehrere Koautoren mehrere Bücher geschrieben haben.

Da Datenbanken aus Tabellen aufgebaut sind, müssen Sie eine Gruppe von Tabellen erstellen, in denen die Daten optimal gespeichert werden können. Dabei enthält jede Zelle der Tabelle eine einzige Sicht. Diese Aufgabe kann auf unterschiedliche Arten ausgeführt werden. Als Datenbankentwickler sind Sie dafür verantwortlich, die optimale Tabellengruppe für Ihre Daten zu entwerfen.

Sie können zum Beispiel eine einzelne Tabelle erstellen, die mehrere Zeilen und Spalten umfasst, in denen alle verfügbaren Informationen gespeichert werden können. Bei diesem Verfahren kommt es allerdings dazu, dass bestimmte Informationen wiederholt aufgeführt sind. Darüber hinaus ist die Dateneingabe und die Datenpflege bei diesem Verfahren zeitaufwendig und fehleranfällig. Im Gegensatz zum Einzeltabellenentwurf ermöglicht Ihnen eine *relationale Datenbank* die Verwendung mehrerer einfacher Tabellen. Hierdurch kann die Datenredundanz reduziert und es können Probleme vermieden werden, die sich durch sehr große und schwierig zu verwaltende Tabellen ergeben. In einer relationalen Datenbank müssen Tabellen die Informationen zu einem einzigen Entitätstyp enthalten.

Darüber hinaus muss in einer relationalen Datenbank die Datenintegrität gewährleistet werden, wenn mehrere Benutzer auf die Daten zugreifen oder sie ändern. Immer wenn Daten gemeinsam verwendet werden, muss die Richtigkeit der Werte innerhalb von Datenbanktabellen sichergestellt werden.

Mögliche Operationen:

- Mithilfe von Isolationsstufen können Sie festlegen, wie Daten gesperrt oder von anderen Prozessen isoliert werden, während auf die Daten zugegriffen wird.
- Sie können Daten schützen und Beziehungen zwischen Daten herstellen, indem Sie Integritätsbedingungen zur Umsetzung von Geschäftsregeln definieren.

- Sie können Trigger erstellen, die eine komplexe, tabellenübergreifende Datenprüfung durchführen können.
- Sie können eine Recoverystrategie implementieren, um Daten so zu schützen, dass sie in einem konsistenten Zustand wiederhergestellt werden können.

Der Datenbankentwurf ist weitaus komplexer als dies hier beschrieben werden kann, und es gibt viele Faktoren, die berücksichtigt werden müssen. Hierzu gehören beispielsweise die Speicherplatzanforderungen, die zu verwendenden Schlüssel, Indizes und Integritätsbedingungen sowie Sicherheitsaspekte und Fragen der Zugriffsberechtigung usw. Einige dieser Informationen finden Sie im DB2 Information Center und in den zahlreichen DB2-Dokumentationen, die zu diesem Thema zur Verfügung stehen.

Empfohlene Dateisysteme

DB2-Datenbanken können in vielen Dateisystemen ausgeführt werden, die von den Plattformen unterstützt werden, auf denen das DB2-Produkt eingesetzt wird.

Welche Dateisysteme verwendet werden können, hängt davon ab, ob Sie mit IBM DB2 pureScale Feature arbeiten möchten.

- „DB2-Umgebungen ohne DB2 pureScale Feature"
- „DB2 pureScale-Umgebungen" auf Seite 112

DB2-Umgebungen ohne DB2 pureScale Feature

IBM empfiehlt die in Tabelle 8 aufgeführten Dateisysteme für DB2 for Linux, UNIX and Windows.

Tabelle 8. Für DB2 for Linux, UNIX and Windows empfohlene Dateisysteme

Plattform	Betriebssystem	Empfohlene Dateisysteme
Linux	Red Hat Enterprise Linux (RHEL)	• IBM General Parallel File System[1] (GPFS) • ext3 • VERITAS File System (VxFS) • Network File System (NFS[2]) mit IBM N-series • Network File System (NFS[2]) mit Network Appliance-Filtern
	SUSE Linux Enterprise Server (SLES)	• GPFS • ext3 • VERITAS File System (VxFS) • NFS[2] mit IBM N-series • NFS[2] mit Network Appliance-Filtern
UNIX	AIX	• GPFS • Enhanced Journaled File System (JFS2) • NFS[2] mit IBM N-series • NFS[2] mit Network Appliance-Filtern • VxFS
	HP-UX	• HP JFS[3] (VxFS)
	Solaris	• UNIX File System (UFS) • ZFS • VxFS
Windows	Alle Windows-Produkte	NTFS

Tabelle 8. Für DB2 for Linux, UNIX and Windows empfohlene Dateisysteme (Forts.)

Plattform	Betriebssystem	Empfohlene Dateisysteme
Anmerkungen: [1] Unter http://www-03.ibm.com/systems/clusters/software/gpfs/index.html finden Sie weitere Informationen zu GPFS. [2] Unter http://www-01.ibm.com/support/docview.wss?uid=swg21169938 finden Sie detaillierte Informationen dazu, welche NFS-Versionen für die Verwendung unter verschiedenen Betriebssystemen überprüft werden. [3] HP JFS unter HP-UX ist eine OEM-Version von VxFS.		

DB2 pureScale-Umgebungen

Für DB2 pureScale Feature ist IBM General Parallel File System (GPFS) erforderlich. Dieses Dateisystem wird gegebenenfalls vom Installationsprogramm für DB2 pureScale Feature automatisch installiert und konfiguriert. Weitere Informationen zu der erforderlichen speziellen Version von GPFS finden Sie bei den Installationsvoraussetzungen. Weitere Informationen zu GPFS finden Sie unter http://www-03.ibm.com/systems/clusters/software/gpfs/index.html.

Verzeichnisse und Dateien einer Datenbank

Beim Erstellen einer Datenbank werden zugehörige Informationen einschließlich Standardinformationen in einer Verzeichnishierarchie gespeichert.

Die hierarchische Verzeichnisstruktur wird automatisch für Sie erstellt. Sie können die Position der Struktur angeben, indem Sie einen Verzeichnispfad oder ein Laufwerk für den Befehl **CREATE DATABASE** angeben. Wenn Sie keine Position angeben, wird eine Standardposition verwendet.

In dem Verzeichnis, das Sie im Befehl **CREATE DATABASE** als Datenbankpfad angeben, wird ein Unterverzeichnis erstellt, das den Namen der Instanz verwendet.

Im Unterverzeichnis mit dem Instanznamen wird das *partitionsglobale Verzeichnis* erstellt. Das partitionsglobale Verzeichnis enthält zugehörige globale Informationen für Ihre neue Datenbank. Das partitionsglobale Verzeichnis hat den Namen `NODE`*`xxxx`*`/SQL`*`yyyyy`*, dabei ist *xxxx* die Datenpartitionsnummer und *yyyyy* das Datenbanktoken (mit der Nummer >=1).

Im *partitionsglobalen Verzeichnis* wird das *memberspezifische* Verzeichnis erstellt. Das memberspezifische Verzeichnis enthält lokale Datenbankinformationen. Das memberspezifische Verzeichnis hat den Namen `MEMBER`*`xxxx`*. Dabei ist *xxxx* die Membernummer.

- In einer DB2 pureScale-Umgebung gibt es für jedes Member ein memberspezifisches Verzeichnis mit der Bezeichnung `MEMBER0000`, `MEMBER0001` usw.
- In einer partitionierten Datenbankumgebung müssen die Membernummern Eins-zu-eins-Entsprechungen mit der zugehörigen Partitionsnummer aufweisen. Demzufolge gibt es ein Verzeichnis `NODE`*`xxxx`* pro Member und Partition. Memberspezifische Verzeichnisse weisen stets den Namen 'MEMBERxxxx' auf und befinden sich immer im partitionsglobalen Verzeichnis.
- Eine Enterprise Server Edition-Umgebung wird in einem einzigen Member ausgeführt und sie verfügt über ein memberspezifisches Verzeichnis mit dem Namen `MEMBER0000`.

Partitionsglobales Verzeichnis

Das partitionsglobale Verzeichnis hat den folgenden Pfad: *ihre_instanz*/NODExxxx/SQLxxxxx.

Das partitionsglobale Verzeichnis enthält die folgenden Dateien:

- Dateien für den globalen WRITE TO FILE-Deadlock-Ereignismonitor, die einen relativen oder keinen Pfad angeben.
- Dateien für Tabellenbereichsinformationen.

  Die Dateien SQLSPCS.1 und SQLSPCS.2 enthalten Tabellenbereichsinformationen. Diese Dateien sind identisch und werden zu Backupzwecken erstellt.
- Steuerdateien für Speichergruppe.

  Die Dateien SQLSGF.1 und SQLSGF.2 enthalten zugehörige Speichergruppeninformationen für die Funktion des dynamischen Speichers einer Datenbank. Diese Dateien sind identisch und werden zu Wartungs- und Backupzwecken erstellt. Sie werden für eine Datenbank erstellt, wenn die Datenbank mit dem Befehl **CREATE DATABASE** erstellt wird, oder wenn eine Datenbank ohne dynamischen Speicher auf DB2 Version 10.1 oder auf eine spätere Version aktualisiert wird.
- Containerdateien für den Tabellenbereich für temporäre Tabellen.

  Das Standardverzeichnis für neue Container ist instance/NODExxxx/<db-name>. Diese Dateien werden von jedem Member lokal verwaltet. Die Namen der Tabellenbereichsdateien werden für jedes Member eindeutig definiert, indem die Membernummer in den Dateinamen eingefügt wird (Beispiel: */speicherpfad*/SAMPLEDB/T0000011/C0000000.TMP/SQL00002.MEMBER0001.TDA).
- Globale Konfigurationsdatei.

  Die Datenbankkonfigurationsparameter in der globalen Konfigurationsdatei SQLDBCONF verweisen auf einzelne, gemeinsam genutzte Ressourcen, die innerhalb der Datenbank konsistent bleiben müssen. Editieren Sie diese Datei nicht. Verwenden Sie zum Ändern von Konfigurationsparametern die Befehle **UPDATE DATABASE CONFIGURATION** und **RESET DATABASE CONFIGURATION**.
- Protokolldateien.

  Die Protokolldatei DB2RHIST.ASC und ihre Backupkopie DB2RHIST.BAK enthalten Protokollinformationen über Backups, Restores, Ladeoperationen für Tabellen, Reorganisationen von Tabellen, Änderungen an Tabellenbereichen und andere Änderungen an einer Datenbank.

  Die Datei DB2TSCHG.HIS enthält ein Protokoll über Tabellenbereichsänderungen auf Protokolldateiebene. Für jede Protokolldatei enthält die Datei DB2TSCHG.HIS Informationen, die bei der Ermittlung der von der Protokolldatei betroffenen Tabellenbereiche helfen. Die Funktion zur Recovery von Tabellenbereichen verwendet Informationen aus dieser Datei, um festzustellen, welche Protokolldateien bei einer Tabellenbereichsrecovery zu verarbeiten sind. Sie können den Inhalt von Protokolldateien in einem Texteditor untersuchen.
- Protokollbezogene Dateien.

  Die globalen Protokollsteuerdateien SQLOGCTL.GLFH.1 und SQLOGCTL.GLFH.2 enthalten Recoveryinformationen auf Datenbankebene (z. B. Informationen über das Hinzufügen neuer Member, während die Datenbank offline ist und zum Verwalten einer gemeinsamen Protokollkette zwischen Membern. Die Protokolldateien selbst werden in Verzeichnissen mit der Bezeichnung LOGSTREAM*xxxx* (ein Verzeichnis für jedes Member) im partitionsglobalen Verzeichnis gespeichert.
- Sperrdateien.

Mithilfe der Sperrdateien der Instanzdatenbank (SQLINSLK und SQLTMPLK) kann sichergestellt werden, dass eine Datenbank nur von einer Instanz des Datenbankmanagers verwendet wird.

- Container für dynamischen Speicher.

Member-spezifisches Verzeichnis

Das memberspezifische Verzeichnis hat den folgenden Pfad: /NODE*xxxx*/SQL*xxxx*/MEMBER*xxxx*.

Das Verzeichnis enthält Objekte, die der ersten erstellten Datenbank zugeordnet sind. Nachfolgend erstellte Datenbanken erhalten höhere Nummern: SQL00002 usw. Diese Unterverzeichnisse unterscheiden Datenbanken, die in dieser Instanz in dem Verzeichnis erstellt werden, das Sie im Befehl **CREATE DATABASE** angegeben haben.

Das Datenbankverzeichnis enthält die folgenden Dateien:

- Dateien für Pufferpoolinformationen.

  Die Dateien SQLBP.1 und SQLBP.2 enthalten Pufferpoolinformationen. Diese Dateien sind identisch und werden zu Backupzwecken erstellt.
- Lokale Ereignismonitordateien.
- Protokollbezogene Dateien.

  Die Protokollsteuerdateien, d. h. SQLOGCTL.LFH.1, die zugehörige Spiegelkopie SQLOGCTL.LFH.2 und SQLOGMIR.LFH, enthalten Informationen zu den aktiven Protokollen. In einer DB2 pureScale-Umgebung besitzt jedes Member einen eigenen Protokolldatenstrom und eine eigene Gruppe lokaler LFH-Dateien, die in den einzelnen Member-spezifischen Verzeichnissen gespeichert werden.

  Tipp: Ordnen Sie das Protokollunterverzeichnis Datenträgern zu, die Sie nicht für Daten verwenden. Auf diese Weise können Datenträgerprobleme auf die Daten oder die Protokolle begrenzt werden. Dies kann zudem einen deutlichen Vorteil für die Leistung mit sich bringen, da die Protokolldateien und die Datenbankcontainer nicht um die Bewegung derselben Plattenschreib-/leseköpfe konkurrieren. Ändern Sie die Position des Protokollunterverzeichnisses mithilfe des Datenbankkonfigurationsparameters **newlogpath**.
- Die lokale Konfigurationsdatei.

  Die lokale Datei SQLDBCONF enthält Datenbankkonfigurationsdaten. Editieren Sie diese Datei nicht. Verwenden Sie zum Ändern von Konfigurationsparametern die Befehle **UPDATE DATABASE CONFIGURATION** und **RESET DATABASE CONFIGURATION**.

Bei der Erstellung einer Datenbank wird gleichzeitig auch ein detaillierter Ereignismonitor für Deadlocks erstellt. In einer Enterprise Server Edition-Umgebung und in partitionierten Datenbankumgebungen werden die Dateien des detaillierten Ereignismonitors für Deadlocks im Datenbankverzeichnis auf dem Katalogknoten gespeichert. In einer DB2 pureScale-Umgebung werden die Dateien des detaillierten Ereignismonitors für Deadlocks im partitionsglobalen Verzeichnis gespeichert. Wenn der Ereignismonitor die für ihn festgelegte maximale Anzahl von Dateien zur Ausgabe erreicht, wird er inaktiviert und eine Nachricht wird in das Benachrichtigungsprotokoll geschrieben. Dadurch wird verhindert, dass der Ereignismonitor zu viel Plattenspeicherplatz belegt. Durch das Entfernen von Ausgabedateien, die nicht mehr benötigt werden, kann der Ereignismonitor bei der nächsten Datenbankaktivierung erneut aktiviert werden.

Weitere Informationen für SMS-Datenbankverzeichnisse in Datenbanken mit nicht dynamischem Speicher

In Datenbanken ohne dynamischen Speicher enthalten die SQLT*-Unterverzeichnisse die SMS-Standardtabellenbereiche (SMS = System Managed Space):

- Das Unterverzeichnis `SQLT0000.0` enthält den Katalogtabellenbereich mit den Systemkatalogtabellen.
- Das Unterverzeichnis `SQLT0001.0` enthält den Standardtabellenbereich für temporäre Tabellen.
- Das Unterverzeichnis `SQLT0002.0` enthält den Standardtabellenbereich für Benutzerdaten.

In jedem Unterverzeichnis bzw. jedem Container wird eine Datei mit dem Namen `SQLTAG.NAM` erstellt. Diese Datei markiert das betreffende Unterverzeichnis als im Gebrauch, sodass bei späteren Erstellungsoperationen für Tabellenbereiche nicht versucht wird, diese Unterverzeichnisse zu verwenden.

Darüber hinaus werden in einer Datei mit dem Namen `SQL*.DAT` Informationen zu jeder Tabelle gespeichert, die das Unterverzeichnis bzw. der Container enthält. Der Stern (*) wird durch eine eindeutige Folge von Ziffern ersetzt, die jede Tabelle identifiziert. Für jede Datei `SQL*.DAT` können je nach Tabellentyp, Reorganisationsstatus der Tabelle bzw. nach Vorhandensein von Indizes, LOB- oder LONG-Felder für die Tabelle eine oder mehrere der folgenden Dateien vorhanden sein:

- `SQL*.BKM` (enthält bei einer MDC- oder ITC-Tabelle Blockzuordnungsinformationen)
- `SQL*.LF` (enthält Daten vom Typ LONG VARCHAR oder LONG VARGRAPHIC)
- `SQL*.LB` (enthält Daten der Typen BLOB, CLOB oder DBCLOB)
- `SQL*.XDA` (enthält XML-Daten)
- `SQL*.LBA` (enthält Informationen zu zugeordnetem und freiem Speicherbereich für `SQL*.LB`-Dateien)
- `SQL*.INX` (enthält Indextabellendaten)
- `SQL*.IN1` (enthält Indextabellendaten)
- `SQL*.DTR` (enthält temporäre Daten für eine Reorganisation einer `SQL*.DAT`-Datei)
- `SQL*.LFR` (enthält temporäre Daten für eine Reorganisation einer `SQL*.LF`-Datei)
- `SQL*.RLB` (enthält temporäre Daten für eine Reorganisation einer `SQL*.LB`-Datei)
- `SQL*.RBA` (enthält temporäre Daten für eine Reorganisation einer `SQL*.LBA`-Datei)

Datenbankkonfigurationsdatei

Für jede Datenbank wird eine *Datenbankkonfigurationsdatei* erstellt. Diese Datei hat vor Version 8.2 den Namen SQLDBCON und ab Version 8.2 den Namen SQLDBCONF. Diese Datei wird automatisch für Sie erstellt.

Diese Datei enthält Werte für verschiedene *Konfigurationsparameter*, die sich auf die Verwendung der Datenbank auswirken, wie die folgenden:

- Parameter, die bei der Erstellung der Datenbank angegeben oder verwendet werden (z. B. Codepage der Datenbank, Sortierfolge, DB2-Datenbank-Release-Level)
- Parameter, die den aktuellen Status der Datenbank anzeigen (z. B. Anzeiger für Backup anstehend, Datenbankkonsistenz, aktualisierende Recovery anstehend)
- Parameter, die die Menge an Systemressourcen definieren, die für den Betrieb der Datenbank verwendet werden sollen (z. B. Pufferpoolgröße, Datenbankprotokollierung, Sortierspeichergröße)

Anmerkung: Wenn Sie db2system oder SQLDBCONF (vor Version 8.2) bzw. SQLDBCONF (ab Version 8.2) mit anderen als den vom DB2-Datenbankmanager bereitgestellten Methoden editieren, können Sie die Datenbank unbrauchbar machen. Aus diesem Grund sollten Sie diese Dateien nicht mit anderen als den dokumentierten und vom Datenbankmanager unterstützten Methoden ändern.

Hinweis zur Leistung: Viele Konfigurationsparameter verfügen zwar über Standardwerte, müssen aber möglicherweise aktualisiert werden, um eine optimale Leistung für Ihre Datenbank zu erzielen. Standardmäßig wird der Konfigurationsadvisor während der Ausführung des Befehls **CREATE DATABASE** aufgerufen, sodass die Anfangswerte für einige der Parameter für Ihre Umgebung bereits konfiguriert sind.

Bei Mehrpartitionsdatenbanken: Wenn Sie eine Datenbank haben, die auf mehr als eine Datenbankpartition verteilt ist, sollte die Konfigurationsdatei in allen Datenbankpartitionen die gleiche sein. Diese Konsistenz ist erforderlich, da der Abfragecompiler verteilte SQL-Anweisungen anhand der Informationen in der Konfigurationsdatei des lokalen Knotens kompiliert und einen Zugriffsplan erstellt, der die Anforderungen der jeweiligen SQL-Anweisung erfüllt. Wenn sich verschiedene Konfigurationsdateien in den Datenbankpartitionen befinden, kann dies je nachdem, in welcher Datenbankpartition die Anweisung vorbereitet wird, zu verschiedenen Zugriffsplänen führen.

Knotenverzeichnis

Der Datenbankmanager erstellt das *Knotenverzeichnis*, wenn die erste Datenbankpartition katalogisiert wird.

Zum Katalogisieren einer Datenbankpartition wird der Befehl **CATALOG NODE** verwendet.

Anmerkung: In einer DB2 pureScale-Umgebung braucht der Befehl CATALOG NODE nicht ausgeführt zu werden, weil DB2 pureScale Feature als einzelne Partition fungiert.

Zum Auflisten des Inhalts des lokalen Knotenverzeichnisses wird der Befehl **LIST NODE DIRECTORY** verwendet.

Das Knotenverzeichnis wird auf jedem Datenbankclient erstellt und verwaltet. Das Verzeichnis enthält einen Eintrag für jeden fernen Datenbankpartitionsserver, der mindestens eine Datenbank enthält, auf die der Client zugreifen kann. Der DB2-Client verwendet die Informationen über den Kommunikationsendpunkt im Knotenverzeichnis jedes Mal, wenn eine Datenbankverbindung oder eine Instanzverbindung (mit ATTACH) angefordert wird.

Die Einträge im Verzeichnis enthalten außerdem Informationen zum Typ des Übertragungsprotokolls, das für die Kommunikation zwischen dem Client und der fernen Datenbankpartition verwendet werden soll. Durch das Katalogisieren einer lokalen Datenbankpartition wird ein Aliasname für eine Instanz erstellt, die sich auf demselben Computer befindet.

Lokales Datenbankverzeichnis

In jedem Pfad (bzw. „Laufwerk" unter Windows-Betriebssystemen), in dem eine Datenbank definiert wurde, ist eine Datei für das *lokale Datenbankverzeichnis* vorhanden. Dieses Verzeichnis enthält einen Eintrag für jede Datenbank, auf die von dieser Position aus zugegriffen werden kann.

Ein Eintrag enthält folgende Daten:

- Den Datenbanknamen, der mit dem Befehl **CREATE DATABASE** angegeben wurde
- Den Aliasnamen der Datenbank (dieser Name ist mit dem Datenbanknamen identisch, wenn kein Aliasname angegeben wird)
- Einen Kommentar zur Datenbank, der mit dem Befehl **CREATE DATABASE** angegeben wurde
- Den Namen des Stammverzeichnisses für die Datenbank
- Andere Systeminformationen

Systemdatenbankverzeichnis

Für jede Instanz des Datenbankmanagers ist eine Datei mit einem *Systemdatenbankverzeichnis* vorhanden, die für jede Datenbank, die für dieser Instanz katalogisiert wurde, einen separaten Eintrag enthält.

Datenbanken werden implizit katalogisiert, wenn der Befehl **CREATE DATABASE** ausgegeben wird. Sie können auch explizit mit dem Befehl **CATALOG DATABASE** katalogisiert werden.

Für jede erstellte Datenbank wird dem Verzeichnis ein Eintrag hinzugefügt, der folgende Informationen enthält:

- Den Datenbanknamen, der mit dem Befehl **CREATE DATABASE** angegeben wurde
- Den Aliasnamen der Datenbank (dieser Name ist mit dem Datenbanknamen identisch, wenn kein Aliasname angegeben wird)
- Den Kommentar zur Datenbank, der mit dem Befehl **CREATE DATABASE** angegeben wurde
- Die Speicherposition des `lokalen Datenbankverzeichnisses`
- Einen Anzeiger, dass die Datenbank *indirekt* ist, d. h., dass sich die Datenbank in der aktuellen Datenbankmanagerinstanz befindet
- Andere Systeminformationen

Auf UNIX-Plattformen und in einer Umgebung mit partitionierten Datenbanken müssen Sie sicherstellen, dass alle Datenbankpartitionen jederzeit auf dieselbe Datei mit dem Systemdatenbankverzeichnis (`sqldbdir`) im Unterverzeichnis `sqldbdir` des Ausgangsverzeichnisses für die Instanz zugreifen. Unvorhersehbare Fehler können auftreten, wenn entweder das Systemdatenbankverzeichnis oder die Systemintentionsdatei `sqldbins` im selben Unterverzeichnis `sqldbdir` symbolische Verbindungen zu einer anderen Datei darstellen, die sich in einem gemeinsam benutzten Dateisystem befindet.

Erstellen von Knotenkonfigurationsdateien

Wenn Ihre Datenbank in einer partitionierten Datenbankumgebung arbeiten soll, müssen Sie eine Knotenkonfigurationsdatei mit dem Namen `db2nodes.cfg` erstellen.

Informationen zu diesem Vorgang

Um eine Datenbankpartitionierung zu ermöglichen, muss sich die Datei `db2nodes.cfg` im Unterverzeichnis `sqllib` des Ausgangsverzeichnisses (Home) für die Instanz befinden, bevor Sie den Datenbankmanager starten. Diese Datei enthält Konfigurationsdaten für alle Datenbankpartitionen einer Instanz und wird von allen Datenbankpartitionen für diese Instanz gemeinsam genutzt.

Hinweise für Windows

Wenn Sie DB2 Enterprise Server Edition unter Windows verwenden, wird die Knotenkonfigurationsdatei beim Erstellen der Instanz erstellt. Sie sollten nicht versuchen, eine Knotenkonfigurationsdatei manuell zu erstellen oder manuell zu ändern. Mit Hilfe des Befehls **db2ncrt** können Sie einer Instanz einen Datenbankpartitionsserver hinzufügen. Mit Hilfe des Befehls **db2ndrop** können Sie einen Datenbankpartitionsserver aus einer Instanz löschen. Mit Hilfe des Befehls **db2nchg** können Sie die Konfiguration der Datenbankpartitionsserver ändern. Dies umfasst das Versetzen des Datenbankpartitionsservers von einem Computer auf einen anderen, das Ändern des TCP/IP-Hostnamens oder das Auswählen eines anderen logischen Portnamens oder Netznamens.

Anmerkung: Erstellen Sie keine anderen als die vom Datenbankmanager erstellten Dateien oder Verzeichnisse unter dem Unterverzeichnis `sqllib`, um Datenverluste zu vermeiden, wenn eine Instanz gelöscht wird. Es gibt jedoch zwei Ausnahmen. Wenn Ihr System gespeicherte Prozeduren unterstützt, stellen Sie die gespeicherten Prozeduranwendungen in das Unterverzeichnis 'function' im Unterverzeichnis `sqllib`. Die andere Ausnahme betrifft eventuell erstellte benutzerdefinierte Funktionen (UDFs). Benutzerdefinierte Funktionen können im selben Verzeichnis gespeichert werden.

Die Datei enthält eine Zeile für jede Datenbankpartition, die zu einer Instanz gehört. Jede Zeile hat folgendes Format:

dbpartitions-
nummer hostname [*logischer-port* [*netzname*]]

Die Token einer Zeile sind durch Leerzeichen voneinander getrennt. Die Variablen sind:

dbpartitionsnummer
Die Datenbankpartitionsnummer (mögliche Werte: `0 - 999`) definiert eine Datenbankpartition eindeutig. Datenbankpartitionsnummern müssen in aufsteigender Reihenfolge angegeben sein. Es dürfen Sprünge in der Folge der Nummern auftreten.

Wenn eine Datenbankpartitionsnummer einmal zugeordnet ist, kann sie nicht mehr geändert werden. (Ansonsten könnten die Informationen in der Verteilungszuordnung, die bestimmt, wie Daten verteilt werden, inkonsistent werden.)

Wenn Sie eine Datenbankpartition löschen, kann ihre Datenbankpartitionsnummer für jede neue Datenbankpartition, die Sie hinzufügen, wieder verwendet werden.

Die Datenbankpartitionsnummer wird zur Generierung eines Datenbankpartitionsnamens im Datenbankverzeichnis verwendet. Er hat folgendes Format:

`NODE` *nnnn*

Die Ziffernfolge *nnnn* ist die Datenbankpartitionsnummer, die links mit Nullen aufgefüllt wird. Diese Datenbankpartitionsnummer wird außerdem durch die Befehle **CREATE DATABASE** und **DROP DATABASE** verwendet.

hostname
Der Hostname der IP-Adresse für die partitionsübergreifende Kommunikation. Verwenden Sie den vollständig qualifizierten Namen für den Hostnamen. In der Datei `/etc/hosts` sollte ebenfalls der vollständig qualifizierte Name verwendet werden. Wenn der vollständig qualifizierte Name in der

Datei db2nodes.cfg und in der Datei /etc/hosts nicht verwendet wird, empfangen Sie eventuell die Fehlernachricht SQL30082N RC=3.

(Es gibt eine Ausnahme, wenn der Netzname angegeben wird. In diesem Fall wird der Netzname für den Großteil der Kommunikation verwendet, während der Hostname nur für die Befehle **db2start**, **db2stop** und **db2_all** verwendet wird.)

logischer-port
Dieser Parameter ist optional und gibt die logische Portnummer für die Datenbankpartition an. Diese Nummer wird mit dem Instanznamen des Datenbankmanagers verwendet, um einen TCP/IP-Servicenamenseintrag in der Datei etc/services anzugeben.

Die Kombination aus IP-Adresse und logischem Port wird als allgemein bekannte Adresse verwendet und muss für alle Anwendungen eindeutig sein, um die Verbindungen zur Kommunikation zwischen Datenbankpartitionen zu unterstützen.

Für jeden Hostnamen muss ein *logischer-port* entweder 0 (null) oder leer sein (was standardmäßig dem Wert 0 entspricht). Die Datenbankpartition, der dieser *logische-port* zugeordnet ist, ist der Standardknoten auf dem Host, zum dem Clients die Verbindung herstellen. Dieses Verhalten kann mithilfe der Umgebungsvariablen **DB2NODE** im Script **db2profile** oder mit der API sqlesetc() überschrieben werden.

netzname
Dieser Parameter ist optional und wird zur Unterstützung eines Hosts verwendet, der über mehr als eine aktive TCP/IP-Schnittstelle verfügt, von denen jede ihren eigenen Hostnamen hat.

Das folgende Beispiel zeigt eine mögliche Knotenkonfigurationsdatei für ein System, auf dem SP2EN1 mehrere TCP/IP-Schnittstellen und zwei logische Datenbankpartitionen hat und SP2SW1 als DB2-Datenbankschnittstelle verwendet. Es zeigt außerdem die Datenbankpartitionsnummern, beginnend bei 1 (und nicht bei 0), sowie einen Sprung in der Folge der *dbpartitionsnummern*:

Tabelle 9. Eine Tabelle mit einem Beispiel für Datenbankpartitionsnummern

dbpartitions-nummer	*hostname*	*logischer-port*	*netzname*
1	SP2EN1.mach1.xxx.com	0	SP2SW1
2	SP2EN1.mach1.xxx.com	1	SP2SW1
4	SP2EN2.mach1.xxx.com	0	
5	SP2EN3.mach1.xxx.com		

Die Datei db2nodes.cfg kann mit einem beliebigen Editor aktualisiert werden. (Ausnahme: Unter Windows sollte kein Editor verwendet werden.) Sie müssen jedoch sorgfältig auf den Schutz der Integrität der Daten in der Datei achten, da die Datenbankpartitionierung erfordert, dass die Knotenkonfigurationsdatei gesperrt wird, wenn der Befehl **START DBM** ausgeführt wird, und entsperrt wird, wenn der Befehl **STOP DBM** den Datenbankmanager beendet hat. Der Befehl **START DBM** kann die Datei bei Bedarf aktualisieren, während sie gesperrt ist. Sie können beispielsweise den Befehl **START DBM** mit der Option **RESTART** oder der Option **ADD DBPARTITIONNUM** ausführen.

Anmerkung: Wenn der Befehl **STOP DBM** nicht erfolgreich ausgeführt und die Knotenkonfigurationsdatei nicht entsperrt wird, führen Sie den Befehl **STOP DBM FORCE** aus, um sie zu entsperren.

Ändern der Knoten- und Datenbankkonfigurationsdatei

Führen Sie zum Aktualisieren der Datenbankkonfigurationsdatei den Befehl **AUTOCONFIGURE** mit den entsprechenden Optionen aus.

Informationen zu diesem Vorgang

Der Konfigurationsadvisor hilft Ihnen bei der Optimierung der Leistung und der ausgewogenen Verteilung des Speicherbedarfs für eine einzelne Datenbank pro Instanz und macht Vorschläge zur Änderung von Konfigurationsparametern und Empfehlungen zu geeigneten Werten.

Wenn Sie Änderungen an bestimmten Datenbankpartitionsgruppen (Hinzufügen oder Löschen von Datenbankpartitionen oder Versetzen vorhandener Datenbankpartitionen) planen, muss die Knotenkonfigurationsdatei aktualisiert werden. Überprüfen Sie die Werte der Konfigurationsparameter, wenn Sie Änderungen an der Datenbank planen. Einige Werte können von Zeit zu Zeit im Rahmen der laufenden Änderungen an der Datenbank je nach Benutzung angepasst werden.

Anmerkung: Wenn Sie Parameter ändern, werden die entsprechenden Werte erst dann aktualisiert, wenn die folgenden Ereignisse stattfinden:

- Bei Datenbankparametern, wenn die erste neue Verbindung zur Datenbank hergestellt wird, nachdem alle Anwendungen getrennt waren.
- Bei Datenbankmanagerparametern, wenn die Instanz das nächste Mal gestoppt und wieder gestartet wird.

In den meisten Fällen führen die vom Konfigurationsadvisor empfohlenen Werte zu einer besseren Leistung als die Standardwerte, weil sie auf den Informationen zur Auslastung Ihres Systems und zu Ihrem speziellen Server basieren. Beachten Sie jedoch, dass die Werte die Leistung Ihres Datenbanksystems verbessern, aber nicht unbedingt optimieren können. Sie sollten deshalb als Ausgangspunkt für weitere Optimierungsmaßnahmen verstanden werden.

In Version 9.1 wird der Konfigurationsadvisor automatisch aufgerufen, wenn Sie eine Datenbank erstellen. Zur Inaktivierung dieser Funktion oder zur expliziten Aktivierung dieser Funktion verwenden Sie den Befehl **db2set**, bevor Sie die Datenbank erstellen. Beispiele:

```
db2set DB2_ENABLE_AUTOCONFIG_DEFAULT=NO
db2set DB2_ENABLE_AUTOCONFIG_DEFAULT=YES
```

Weitere Funktionen, die standardmäßig aktiviert sind, finden Sie in „Automatische Funktionen" auf Seite 23.

Vorgehensweise

Gehen Sie wie folgt vor, um die Konfigurationsparameter für die Datenbank oder den Datenbankmanager zu ändern:

- Verwenden Sie den Konfigurationsadvisor.
  - Geben Sie den Befehl **AUTOCONFIGURE** in die Befehlszeile ein.
  - Rufen Sie aus einer Clientanwendung heraus die API db2AutoConfig auf.

- Klicken Sie in IBM Data Studio mit der rechten Maustaste auf die Instanz oder die Datenbank, um den Taskassistenten zu öffnen und die Konfigurationsparameter für den Datenbankmanager oder die Datenbank zu ändern.
- Verwenden Sie in der Befehlszeile die Befehle **UPDATE DATABASE MANAGER CONFIGURATION** und **UPDATE DATABASE CONFIGURATION**.

  Geben Sie zum Aktualisieren einzelner Parameter in der Datenbankmanagerkonfiguration beispielsweise Folgendes ein:

  ```
  UPDATE DBM CFG USING konfig_schlüsselwort wert
  ```

  Geben Sie zum Aktualisieren einzelner Parameter in der Datenbankkonfiguration Folgendes ein:

  ```
  UPDATE DB CFG FOR aliasname_der_datenbank
  USING konfig_schlüsselwort wert
  ```

  Sie können eine oder mehrere Kombinationen aus *konfig_schlüsselwort* und *wert* in einem einzigen Befehl aktualisieren.

  Die meisten Änderungen an der Konfigurationsdatei des Datenbankmanagers werden erst wirksam, nachdem Sie in den Speicher geladen wurden. Bei einem Konfigurationsparameter für einen Server wird diese Operation während der Ausführung des Befehls **START DATABASE MANAGER** durchgeführt. Bei einem Konfigurationsparameter für einen Client erfolgt die Ausführung dieser Operation beim Neustart der Anwendung.
- Rufen Sie aus einer Clientanwendung heraus die API db2CfgSet auf.

Nächste Schritte

Zum Anzeigen oder Ausgeben der aktuellen Konfigurationsparameter für den Datenbankmanager können Sie den Befehl **GET DATABASE MANAGER CONFIGURATION** verwenden.

Zugehörige Informationen:

Datenbankrecoveryprotokoll

Ein *Datenbankrecoveryprotokoll* zeichnet Datensätze über alle an einer Datenbank vorgenommenen Änderungen auf, wozu auch das Hinzufügen neuer Tabellen oder das Aktualisieren vorhandener Tabellen gehören.

Dieses Protokoll besteht aus einer Reihe von *Protokollspeicherbereichen*, die jeweils in einer separaten, so genannten *Protokolldatei* enthalten sind.

Das Datenbankrecoveryprotokoll kann verwendet werden, um sicherzustellen, dass eine Datenbank nach einem Fehler (z. B. ein Netzstromausfall im System oder ein Anwendungsfehler) nicht in einem inkonsistenten Zustand bleibt. Im Fall eines Fehlers werden bereits durchgeführte Änderungen, die noch nicht mit COMMIT festgeschrieben wurden, mit ROLLBACK rückgängig gemacht; alle festgeschriebenen Transaktionen, die eventuell noch nicht physisch auf die Platte geschrieben wurden, werden wiederhergestellt. Durch diese Maßnahmen wird die Integrität der Datenbank gewährleistet.

Speicherbedarf für Datenbankobjekte

Das Abschätzen der Größe von Datenbankobjekten ist ein ungenaues Unterfangen. Der Systemaufwand, der durch Datenträgerfragmentierung, freien Speicherbereich und die Verwendung von Spalten variabler Länge bedingt ist, macht eine Abschätzung schwierig, weil es eine große Bandbreite an Möglichkeiten für Spaltentypen und Zeilenlängen gibt.

Erstellen Sie nach dem Abschätzen der Größe Ihrer Datenbank eine Testdatenbank, und füllen Sie sie mit repräsentativen Daten. Verwenden Sie dann das Dienstprogramm db2look, um Datendefinitionsanweisungen für die Datenbank zu generieren.

Bei der Abschätzung der Größe einer Datenbank müssen die Auswirkungen der folgenden Elemente berücksichtigt werden:

- Systemkatalogtabellen
- Benutzertabellendaten
- LF-Daten (LF = Long Field; Langfeld)
- LOB-Daten (LOB = Large Object; großes Objekt)
- XML-Daten
- Indexbereich
- Protokolldateispeicher
- Temporärer Arbeitsspeicherbereich

Berücksichtigen Sie außerdem den Aufwand und den Platzbedarf für folgende Komponenten:

- Datei für das lokale Datenbankverzeichnis
- Datei für das Systemdatenbankverzeichnis
- Bei der Dateiverwaltung entstehender Systemaufwand, den das Betriebssystem benötigt, zum Beispiel:
  - Dateiblockgröße
  - Speicherbereich für die Verzeichnissteuerung

Speicherbedarf für Protokolldateien

Der Speicherplatzbedarf für Protokolldateien variiert abhängig von Ihren Anforderungen sowie von Konfigurationsparametereinstellungen.

Sie benötigen 56 KB Speicherplatz für Protokollsteuerdateien. Darüber hinaus benötigen Sie mindestens ausreichend Speicherplatz für Ihre aktive Protokollkonfiguration, die Sie nach folgender Formel berechnen können:

$$(logprimary + logsecond) \times (logfilsiz + 2) \times 4096$$

Dabei gilt Folgendes:

- `logprimary` ist die Anzahl der primären Protokolldateien, die in der Konfigurationsdatei der Datenbank definiert sind.
- `logsecond` ist die Anzahl der sekundären Protokolldateien, die in der Konfigurationsdatei der Datenbank definiert sind. In dieser Formel kann `logsecond` nicht auf den Wert `-1` gesetzt werden. (Wenn `logsecond` auf den Wert `-1` gesetzt wird, fordern Sie einen unbegrenzten Speicherplatz für aktive Protokolldateien an.)
- `logfilsiz` ist die Anzahl der Seiten in jeder Protokolldatei, die in der Konfigurationsdatei der Datenbank definiert sind.
- 2 ist die Anzahl der Kopfseiten, die für jede Protokolldatei erforderlich ist.
- 4096 ist die Anzahl der Byte einer Seite.

Aktualisierende Recovery

Wenn für die Datenbank die aktualisierende Recovery aktiviert ist, sollten möglicherweise spezielle Speicheranforderungen für Protokolle berücksichtigt werden:

- Ist der Konfigurationsparameter **logarchmeth1** auf LOGRETAIN gesetzt, werden die Protokolldateien im Verzeichnis logpath archiviert. Der Online-Plattenspeicherplatz wird schließlich belegt, sofern Sie die Protokolldateien nicht an eine andere Speicherposition verschieben.
- Ist der Konfigurationsparameter **logarchmeth1** auf USEREXIT, DISK oder VENDOR gesetzt, versetzt ein Benutzerexitprogramm die archivierten Protokolldateien an eine andere Speicherposition. Zusätzlicher Speicherbereich ist weiterhin erforderlich für folgende Dateien:
  - Online archivierte Protokolldateien, die darauf warten, vom Benutzerexitprogramm verschoben zu werden
  - Neue Protokolle, die für künftige Zwecke formatiert werden
- Sie können die Kosten für die Speicherung Ihrer archivierten Protokolldateien verringern, indem Sie für diese Dateien die Komprimierung aktivieren.
  - Wenn beispielsweise der Konfigurationsparameter **logarchmeth1** mit dem Wert DISK, TSM oder VENDOR definiert ist und Sie legen für den Konfigurationsparameter **logarchcompr1** den Wert ON fest, werden archivierte Protokolldateien komprimiert und Sie verringern die Kosten für die Speicherung dieser Dateien. Wenn Sie die Komprimierung dynamisch aktivieren, werden bereits vorhandene archivierte Protokolldateien nicht komprimiert. Bei Aktivierung der Komprimierung wird mit der aktuellen aktiven Protokolldatei begonnen.

Umlaufprotokollierung
Wenn für die Datenbank Umlaufprotokollierung aktiviert ist, liefert das Ergebnis dieser Formel den Betrag des gesamten Speicherplatzes, der für die Protokollierung zugeordnet ist. Das heißt, es wird kein weiterer Speicherplatz zugeordnet und für keine Ihrer Protokolldateien werden Fehler über nicht ausreichenden Plattenspeicherplatz ausgegeben.

Endlosprotokollierung
Wenn für die Datenbank eine unendliche Protokollierung aktiviert ist (d. h., wenn der Konfigurationsparameter **logsecond** auf den Wert -1 gesetzt ist), muss der Konfigurationsparameter **logarchmeth1** auf einen anderen Wert als OFF oder LOGRETAIN gesetzt werden, um die Archivprotokollierung zu aktivieren. Der Datenbankmanager behält mindestens die durch den Konfigurationsparameter **logprimary** definierte Anzahl aktiver Protokolldateien im Protokollpfad bei. Daher dürfen Sie in der obigen Formel für den Konfigurationsparameter **logsecond** nicht den Wert -1 verwenden. Stellen Sie sicher, dass Sie zusätzlichen Speicherplatz bereitstellen, um der durch die Archivierung von Protokolldateien verursachten Verzögerung Rechnung zu tragen.

Spiegeln von Protokollpfaden
Wenn Sie den Protokollpfad spiegeln, müssen Sie den Schätzwert für den Protokolldateispeicherbedarf verdoppeln.

Zurzeit festgeschriebene Daten
Wenn Abfragen den zurzeit festgeschriebenen Wert der Daten zurückgeben, wird mehr Protokollspeicherbereich zur Protokollierung der ersten Aktualisierung einer Datenzeile während einer Transaktion benötigt, wenn der Konfigurationsparameter **cur_commit** nicht auf den Wert DISABLED gesetzt ist. Abhängig vom Umfang der Auslastung kann der Protokollspeicherbereich, der insgesamt verwendet wird, erheblich variieren. Dieses Szenario wirkt sich auf die für eine beliebige Workload erforderlichen Ein-/Ausgaben für die Protokollierung, die Größe des erforderlichen

Speicherbereichs für aktive Protokolle und die Größe des erforderlichen Speicherbereichs für die Protokollarchivierung aus.

Anmerkung: Die Einstellung des Konfigurationsparameters **cur_commit** auf den Wert DISABLED sorgt dafür, dass dasselbe Verhalten wie in früheren Releases erhalten bleibt, und hat keine Änderungen am Protokollspeicherbedarf zur Folge.

LDAP-Verzeichnisservice (Lightweight Directory Access Protocol)

Bei einem Verzeichnisservice handelt es sich um ein Repository mit Ressourceninformationen zu mehreren Systemen und Services innerhalb einer verteilten Umgebung. Er stellt den Client- und Serverzugriff auf diese Ressourcen bereit.

Clients und Server verwenden den Verzeichnisservice in der Regel, um festzustellen, wie sie auf andere Ressourcen zugreifen können. Informationen zu diesen anderen Ressourcen in der verteilten Umgebung müssen in das Repository des Verzeichnisservice eingegeben werden.

Lightweight Directory Access Protocol (LDAP) ist eine dem Branchenstandard entsprechende Methode für den Zugriff auf Verzeichnisservices. Jede Datenbankserverinstanz veröffentlicht Informationen über ihre Existenz auf einem LDAP-Server und stellt dem LDAP-Verzeichnis Datenbankinformationen zur Verfügung, wenn die Datenbanken erstellt werden. Wenn ein Client eine Verbindung zur Datenbank herstellt, können die Kataloginformationen für den Server aus dem LDAP-Verzeichnis abgerufen werden. Die einzelnen Clients müssen die Kataloginformationen nun nicht mehr lokal auf den verschiedenen Computern speichern. Clientanwendungen durchsuchen das LDAP-Verzeichnis nach den erforderlichen Informationen für die Herstellung der Verbindung zur Datenbank.

Anmerkung: Das Veröffentlichen der Datenbankserverinstanz auf dem LDAP-Server ist kein automatischer Prozess, sondern muss vom Administrator manuell ausgeführt werden.

Als Administrator eines DB2-Systems können Sie einen Verzeichnisservice einrichten und verwalten. Der Verzeichnisservice wird für den DB2-Datenbankmanager durch LDAP-Verzeichnisservices verfügbar gemacht. Zur Verwendung von LDAP-Verzeichnisservices muss zunächst ein LDAP-Server vorhanden sein, der von einem DB2-Datenbankmanager unterstützt wird, sodass dort Verzeichnisinformationen gespeichert werden können.

Anmerkung: In einer Domänenumgebung unter Windows steht bereits ein LDAP-Server zur Verfügung, da er in Windows Active Directory integriert ist. Infolgedessen kann jeder Computer, der mit Windows arbeitet, LDAP verwenden.

Ein LDAP-Verzeichnis ist in einer Unternehmensumgebung nützlich, in der es aufgrund einer großen Anzahl von Clients schwierig ist, lokale Verzeichniskataloge auf jedem Client-Computer zu aktualisieren. In einer solchen Situation wird empfohlen, die Verzeichniseinträge auf einem LDAP-Server zu speichern, sodass die Verwaltung von Katalogeinträgen zentral über einen Punkt, d. h. über den LDAP-Server, erfolgen kann.

Erstellen von Datenbanken

Sie erstellen eine Datenbank mit dem Befehl **CREATE DATABASE**. Zur Erstellung einer Datenbank aus einer Clientanwendung heraus rufen Sie die API sqlecrea auf. Alle Datenbanken werden mit der Standardspeichergruppe IBMSTOGROUP erstellt, sofern Sie nichts anderes angeben. Für die Speicherdefinitionen von mit dynamischem Speicher verwalteten Tabellenbereichen werden Speichergruppen verwendet.

Vorbereitende Schritte

Der DB2-Datenbankmanager muss aktiv sein. Verwenden Sie zum Starten des Datenbankmanagers den Befehl **db2start**.

Es ist wichtig, die Datenbank vor der Erstellung zu planen, wobei der Inhalt, das Layout, das potenzielle Wachstum und die Verwendungsweise der Datenbank als Faktoren zu berücksichtigen sind. Nachdem sie erstellt und mit Daten gefüllt wurde, können Änderungen vorgenommen werden.

Die folgenden Datenbankzugriffsrechte werden der Gruppe PUBLIC automatisch erteilt: CREATETAB, BINDADD, CONNECT, IMPLICIT_SCHEMA und SELECT für die Systemkatalogsichten. Wenn jedoch die Option **RESTRICTIVE** angegeben wird, werden der Gruppe PUBLIC keine Zugriffsrechte automatisch erteilt. Weitere Informationen zur Option **RESTRICTIVE** finden Sie in den Informationen zum Befehl **CREATE DATABASE**.

Einschränkungen

- Speicherpfade können nicht unter Verwendung relativer Pfadnamen angegeben werden. Sie müssen absolute Pfadnamen verwenden. Der Speicherpfadname kann bis zu 175 Zeichen lang sein.
- Unter Windows-Betriebssystemen muss der Datenbankpfad ein Laufwerkbuchstabe sein, wenn die Registrierdatenbankvariable **DB2_CREATE_DB_ON_PATH** nicht auf den Wert YES gesetzt ist.
- Wenn Sie keinen Datenbankpfad mit der Klausel **DBPATH ON** des Befehls **CREATE DATABASE** angeben, verwendet der Datenbankmanager den ersten Speicherpfad, der für die Klausel **ON** für den Datenbankpfad angegeben ist. (Wenn diese Klausel unter Windows-Betriebssystemen als Pfad angegeben wird und die Registrierdatenbankvariable **DB2_CREATE_DB_ON_PATH** nicht auf den Wert YES gesetzt ist, empfangen Sie eine Fehlernachricht SQL1052N.) Wenn keine Klausel **ON** angegeben wird, wird die Datenbank im Standarddatenbankpfad erstellt, der in der Konfigurationsdatei des Datenbankmanagers (Parameter **dftdbpath**) angegeben ist. Dieser Pfad wird darüber hinaus auch als Position für den einen Speicherpfad, der der Datenbank zugeordnet ist, verwendet.
- Für partitionierte Datenbanken müssen Sie in jeder Datenbankpartition die gleiche Gruppe von Speicherpfaden verwenden (sofern Sie keine Datenbankpartitionsausdrücke verwenden).
- Datenbankpartitionsausdrücke sind in Datenbankpfaden nicht gültig, unabhängig davon, ob Sie sie explizit mit der Klausel **DBPATH ON** des Befehls **CREATE DATABASE** oder implizit durch einen Datenbankpartitionsausdruck im ersten Speicherpfad angeben.
- Einer Speichergruppe muss mindestens ein Speicherpfad zugeordnet sein.

Anmerkung: Obwohl Sie beim Erstellen einer Datenbank die Klausel **AUTOMATIC STORAGE NO** angeben können, ist die Klausel **AUTOMATIC STORAGE** veraltet und wird in einem zukünftigen Release möglicherweise entfernt.

Informationen zu diesem Vorgang

Beim Erstellen einer Datenbank werden die folgenden Tasks für Sie ausgeführt:

- Einrichten aller Systemkatalogtabellen, die von der Datenbank benötigt werden
- Zuordnen des Datenbankrecoveryprotokolls
- Erstellen der Datenbankkonfigurationsdatei und Definieren der Standardwerte
- Binden der Datenbankdienstprogramme an die Datenbank

Vorgehensweise

- Zur Erstellung einer Datenbank aus einer Clientanwendung heraus rufen Sie die API sqlecrea auf.
- Zum Erstellen einer Datenbank über den Befehlszeilenprozessor müssen Sie den Befehl **CREATE DATABASE** absetzen.

  Mit dem folgenden Befehl wird zum Beispiel eine Datenbank namens PERSON1 an der Standardspeicherposition mit dem zugeordneten Kommentar "Personnel DB for BSchiefer Co" erstellt:

  ```
  CREATE DATABASE person1
    WITH "Personnel DB for BSchiefer Co"
  ```

- Zum Erstellen einer Datenbank mithilfe von IBM Data Studio klicken Sie mit der rechten Maustaste auf die Instanz, in der Sie die Datenbank erstellen wollen, und wählen Sie für die Erstellung den Taskassistenten aus.

Beispiel

Beispiel 1: Erstellen einer Datenbank unter einem UNIX- oder Linux-Betriebssystem:

Mit dem folgenden Befehl wird eine Datenbank mit dem Namen TESTDB1 im Pfad /DPATH1 unter Verwendung der Speicherpfade /DATA1 und /DATA2 erstellt, die für die Standardspeichergruppe IBMSTOGROUP definiert sind:

```
CREATE DATABASE TESTDB1 ON '/DATA1','/DATA2' DBPATH ON '/DPATH1'
```

Beispiel 2: Erstellen einer Datenbank unter einem Windows-Betriebssystem mit Angabe von Speicher- und Datenbankpfad:

Mit dem folgenden Befehl wird eine Datenbank mit dem Namen TESTDB2 auf Laufwerk D: mit Speicher im Pfad E:\DATA erstellt:

```
CREATE DATABASE TESTDB2 ON 'E:\DATA' DBPATH ON 'D:'
```

In diesem Beispiel wird E:\DATA als Speicherpfad, der für die Standardspeichergruppe IBMSTOGROUP definiert ist, und als Datenbankpfad verwendet.

Beispiel 3: Erstellen einer Datenbank unter einem Windows-Betriebssystem nur mit Angabe eines Speicherpfads:

Mit dem folgenden Befehl wird eine Datenbank mit dem Namen TESTDB3 mit Speicher auf Laufwerk F: erstellt:

```
CREATE DATABASE TESTDB3 ON 'F:'
```

In diesem Beispiel wird F: als Speicherpfad, der für die Standardspeichergruppe IBMSTOGROUP definiert ist, und als Datenbankpfad verwendet.

Wenn Sie einen Verzeichnisnamen wie F:\DATA für den Speicherpfad angeben, schlägt der Befehl aus folgenden Gründen fehl:

1. Wenn **DBPATH** nicht angegeben wird, wird der Speicherpfad (in diesem Fall F:\DATA) als Datenbankpfad verwendet.
2. Unter Windows darf der Datenbankpfad nur ein Laufwerkbuchstabe sein (sofern Sie den Standardwert NO für die Registrierdatenbankvariable **DB2_CREATE_DB_ON_PATH** nicht in YES ändern).

Wenn Sie ein Verzeichnis als Speicherpfad unter Windows-Betriebssystemen angeben wollen, müssen Sie auch die Klausel **DBPATH ON** laufwerk angeben, wie in Beispiel 2 gezeigt.

Beispiel 4: Erstellen einer Datenbank unter einem UNIX- oder Linux-Betriebssystem ohne Angabe eines Datenbankpfads:

Mit dem folgenden Befehl wird eine Datenbank mit dem Namen TESTDB4 und Speicher in /DATA1 und /DATA2 erstellt:

```
CREATE DATABASE TESTDB4 ON '/DATA1','/DATA2'
```

In diesem Beispiel werden /DATA1 und /DATA2 als Speicherpfade verwendet, die für die Standardspeichergruppe IBMSTOGROUP definiert sind, wobei /DATA1 gleichzeitig der Datenbankpfad ist.

Nächste Schritte

Konfigurationsadvisor

Der Konfigurationsadvisor hilft Ihnen bei der Optimierung der Leistung und der ausgewogenen Verteilung des Speicherbedarfs für eine einzelne Datenbank pro Instanz, indem er Konfigurationsparameter zur Änderung vorschlägt und geeignete Werte für sie empfiehlt. Der Konfigurationsadvisor wird standardmäßig automatisch aufgerufen, wenn Sie eine Datenbank erstellen.

Sie können diesen Standardwert überschreiben, sodass der Konfigurationsadvisor nicht automatisch aufgerufen wird. Verwenden Sie hierzu eines der folgenden Verfahren:

- Setzen Sie den Befehl **CREATE DATABASE** mit dem Parameter **AUTOCONFIGURE APPLY NONE** ab.
- Setzen Sie die Registrierdatenbankvariable **DB2_ENABLE_AUTOCONFIG_DEFAULT** auf NO:

  ```
  db2set DB2_ENABLE_AUTOCONFIG_DEFAULT=NO
  ```

  Wenn Sie jedoch den Parameter **AUTOCONFIGURE** mithilfe des Befehls **CREATE DATABASE** angeben, wird die Einstellung dieser Registrierdatenbankvariablen ignoriert.

In „Automatische Funktionen" auf Seite 23 finden Sie weitere Funktionen, die standardmäßig aktiviert werden, wenn Sie eine Datenbank erstellen.

Ereignismonitor

Bei der Erstellung einer Datenbank wird gleichzeitig auch ein detaillierter Ereignismonitor für Deadlocks erstellt. Wie bei jedem Monitor erfordert dieser Ereignismonitor zusätzliche Verarbeitungszeit und -ressourcen. Wenn der detaillierte Ereignismonitor für Deadlocks nicht aktiviert sein soll, kann er mit dem folgenden Befehl gelöscht werden:

```
DROP EVENT MONITOR db2detaildeadlock
```

Zur Begrenzung der Größe des Plattenspeicherplatzes, den dieser Ereignismonitor beansprucht, wird der Ereignismonitor inaktiviert und eine Nach-

richt wird in das Protokoll mit Benachrichtigungen für die Systemverwaltung geschrieben, wenn die maximale Anzahl von Ausgabedateien erreicht ist. Durch das Entfernen von Ausgabedateien, die nicht mehr benötigt werden, kann der Ereignismonitor bei der nächsten Datenbankaktivierung erneut aktiviert werden.

Ferne Datenbanken

Sie haben die Möglichkeit, eine Datenbank in einer anderen, möglicherweise fernen Instanz zu erstellen. Um eine Datenbank in einem anderen (fernen) Datenbankpartitionsserver zu erstellen, müssen Sie zuerst eine Verbindung zu diesem Server herstellen. Eine Datenbankverbindung wird während der Verarbeitung mit dem folgenden Befehl temporär hergestellt:

```
CREATE DATABASE datenbankname AT DBPARTITIONNUM optionen
```

In diesem Typ von Umgebung können Sie Verwaltungsoperationen auf Instanzebene an einer anderen Instanz als Ihrer Standardinstanz, einschließlich ferner Instanzen, ausführen. Anweisungen zur Vorgehensweise finden Sie in den Informationen zum Befehl **db2iupdt** (Instanz aktualisieren).

Datenbankcodepages

Standardmäßig werden Datenbanken im codierten Zeichensatz UTF-8 (Unicode) erstellt.

Wenn Sie die Standardcodepage für die Datenbank überschreiben möchten, muss der erforderliche codierte Zeichensatz und das entsprechende Gebiet bei der Erstellung der Datenbank angegeben werden. Informationen zur Angabe des codierten Zeichensatzes und des Gebiets finden Sie in den Informationen zum Befehl **CREATE DATABASE** bzw. zur API sqlecrea.

Zugehörige Informationen:

Konvertieren einer Datenbank mit nicht dynamischem Speicher zur Verwendung von dynamischem Speicher

Sie können eine vorhandene Datenbank mit nicht dynamischem Speicher zur Verwendung von dynamischem Speicher konvertieren, indem Sie mithilfe der Anweisung CREATE STOGROUP die Standardspeichergruppe in der Datenbank definieren.

Vorbereitende Schritte

Sie müssen eine Speicherposition haben, die Sie durch einen Pfad angeben können (unter Windows-Betriebssysteme durch einen Pfad oder einen Laufwerkbuchstaben), der zur Verwendung als Speicherpfad für die Tabellenbereiche mit dynamischem Speicher verfügbar ist.

Einschränkungen

- Nachdem Sie eine Speichergruppe erstellt haben, können Sie nicht alle Speichergruppen für eine Datenbank löschen.
- Nur DMS-Tabellenbereiche können zur Verwendung des dynamischen Speichers konvertiert werden.

Anmerkung: Eine Datenbank kann mit der Klausel AUTOMATIC STORAGE NO erstellt werden; die Klausel AUTOMATIC STORAGE ist jedoch veraltet und wird in zukünftigen Releases möglicherweise nicht mehr enthalten sein.

Informationen zu diesem Vorgang

Datenbanken, die unter Angabe der Klausel AUTOMATIC STORAGE NO des Befehls **CREATE DATABASE** erstellt werden, sind keine Speichergruppen zugeordnet. Stattdessen wird der Speicher den Tabellenbereichen der Datenbank zugeordnet. Beim Definieren einer Speichergruppe für eine Datenbank werden vorhandene Tabellenbereiche nicht automatisch zur Verwendung von dynamischem Speicher konvertiert. Standardmäßig verwenden erst nachfolgend erstellte Tabellenbereiche den dynamischen Speicher. Sie müssen die Anweisung ALTER TABLESPACE verwenden, um vorhandene Tabellenbereiche für die Verwendung des dynamischen Speichers zu konvertieren.

Vorgehensweise

Sie können eine vorhandene Datenbank in eine Datenbank mit dynamischem Speicher konvertieren, indem Sie mithilfe der Anweisung CREATE STOGROUP eine Speichergruppe in dieser Datenbank erstellen.

Verwenden Sie die folgende Anweisung, um eine Speichergruppe in einer Datenbank zu erstellen:

```
CREATE STOGROUP sg ON storagePath
```

Dabei ist *sg* die Speichergruppe und *speicherPfad* der Pfad, der für Tabellenbereiche mit dynamischem Speicher verwendet werden soll.

Beispiel

Beispiel 1: Konvertieren einer Datenbank unter UNIX- oder Linux-Betriebssystemen

Nehmen Sie an, dass die Datenbank EMPLOYEE eine Datenbank mit nicht dynamischem Speicher ist und dass `/data1/as` und `/data2/as` die Pfade sind, die für Tabellenbereiche mit dynamischem Speicher verwendet werden sollen. Zum Konvertieren der Datenbank EMPLOYEE in eine Datenbank mit dynamischem Speicher erstellen Sie eine Speichergruppe mit den Pfaden `/data1/as` und `/data2/as` wie folgt:

```
CREATE STOGROUP sg ON '/data1/as', '/data2/as'
```

Beispiel 2: Konvertieren einer Datenbank unter Windows-Betriebssystemen

Nehmen Sie an, dass die Datenbank SALES eine Datenbank mit nicht dynamischem Speicher ist und dass `F:\DB2DATA` und `G:` die Pfade sind, die für Tabellenbereiche mit dynamischem Speicher verwendet werden sollen. Zum Konvertieren der Datenbank SALES in eine Datenbank mit dynamischem Speicher erstellen Sie eine Speichergruppe mit den Pfaden `F:\DB2DATA` und `G:` wie folgt:

```
CREATE STOGROUP sg ON 'F:\DB2DATA', 'G:'
```

Nächste Schritte

Wenn Sie DMS-Tabellenbereiche haben, die Sie zur Verwendung von dynamischem Speicher konvertieren wollen, verwenden Sie die Anweisung ALTER TABLESPACE mit der Klausel MANAGED BY AUTOMATIC STORAGE, um sie zu ändern. Wenn Sie die Klausel USING STOGROUP nicht angeben, verwendet der Tabellenbereich die Speicherpfade in der angegebenen Standardspeichergruppe.

Nachdem Sie eine Speichergruppe erstellt haben, können Sie mithilfe der Anweisung CREATE TABLESPACE Tabellenbereiche mit dynamischem Speicher erstellen, um Tabellen, Indizes und andere Datenbankobjekte zu speichern.

Auswirkungen auf Restoreoperationen für Datenbanken

Der Befehl **RESTORE DATABASE** dient dazu, eine Datenbank von einem Backup-Image wiederherzustellen.

Während einer Restoreoperation ist es möglich, die Position des Datenbankpfads zu wählen und darüber hinaus die Speicherpfade erneut zu definieren, die den Speichergruppen zugeordnet sind. Der Datenbankpfad und die Speicherpfade werden definiert, indem im Befehl **RESTORE DATABASE** eine Kombination der Klauseln **TO**, **ON** und **DBPATH ON** oder der Befehl **SET STOGROUP PATHS** verwendet wird.

Nachfolgend sind einige Beispiele für gültige **RESTORE**-Befehle aufgeführt:

```
RESTORE DATABASE TEST1
RESTORE DATABASE TEST2 TO X:
RESTORE DATABASE TEST3 DBPATH ON D:
RESTORE DATABASE TEST3 ON /pfad1, /pfad2, /pfad3
RESTORE DATABASE TEST4 ON E:\neupfad1, F:\neupfad2 DBPATH ON D:
```

Wie auch beim Befehl **CREATE DATABASE** extrahiert der Datenbankmanager die beiden folgenden Informationen, die sich auf Speicherpositionen beziehen:

- Den *Datenbankpfad* (in dem der Datenbankmanager eine Reihe von Steuerdateien für die Datenbank speichert)
  - Wenn die Klauseln **TO** oder **DBPATH ON** angegeben werden, geben diese Klauseln den Datenbankpfad an.
  - Wird die Klausel **ON** ohne **DBPATH ON** verwendet, wird der erste in der Klausel **ON** aufgeführte Pfad als Datenbankpfad verwendet (zusätzlich zu seiner Eigenschaft als Speicherpfad).
  - Wird keine der Klauseln **TO**, **ON** oder **DBPATH ON** angegeben, bestimmt der Konfigurationsparameter **dftdbpath** des Datenbankmanagers den Datenbankpfad.

  Anmerkung: Wenn eine Datenbank mit demselben Namen bereits auf der Platte vorhanden ist, wird der Datenbankpfad ignoriert, und die Datenbank wird im selben Verzeichnis gespeichert wie die vorhandene Datenbank.
- Die *Speicherpfade* für jede *Speichergruppe* (in denen der Datenbankmanager Tabellenbereichscontainer mit dynamischem Speicher erstellt)
  - Wenn die Klausel **ON** angegeben wird, werden alle aufgeführten Pfade als Speicherpfade interpretiert und diese Pfade werden anstelle der im Backup-Image gespeicherten Pfade verwendet. Wenn die Datenbank mehrere Speichergruppen enthält, verwendet jede definierte Speichergruppe die neuen Speichergruppenpfade.
  - Bei Verwendung des Befehls **SET STOGROUP PATHS** werden für die angegebene Speichergruppe die bereitgestellten Speicherpfade und nicht die im Backup-Image gespeicherten Speicherpfade verwendet.
  - Wird die Klausel **ON** nicht angegeben und der Befehl **SET STOGROUP PATHS** nicht verwendet, werden an den Speicherpfaden keine Änderungen vorgenommen (die im Backup-Image gespeicherten Speicherpfade werden beibehalten).

Zur Verdeutlichung dieses Konzepts werden die fünf oben gezeigten Beispiele für den Befehl **RESTORE** in der folgenden Tabelle mit den entsprechenden Speicherpfaden dargestellt:

Tabelle 10. Auswirkungen des Befehls RESTORE auf Datenbank- und Speicherpfade

Befehl **RESTORE DATABASE**	**Auf der Platte ist keine Datenbank mit demselben Namen vorhanden**		**Auf der Platte ist eine Datenbank mit demselben Namen vorhanden**	
	Datenbankpfad	**Speicherpfade**	**Datenbankpfad**	**Speicherpfade**
`RESTORE DATABASE TEST1`	**dftdbpath**	Die im Backup-Image definierten Speicherpfade werden verwendet.	Der Datenbankpfad der vorhandenen Datenbank wird verwendet.	Die im Backup-Image definierten Speicherpfade werden verwendet.
`RESTORE DATABASE TEST2 TO X:`	`X:`	Die im Backup-Image definierten Speicherpfade werden verwendet.	Der Datenbankpfad der vorhandenen Datenbank wird verwendet.	Die im Backup-Image definierten Speicherpfade werden verwendet.
`RESTORE DATABASE TEST3 DBPATH ON /db2/databases`	`/db2/databases`	Die im Backup-Image definierten Speicherpfade werden verwendet.	Der Datenbankpfad der vorhandenen Datenbank wird verwendet.	Die im Backup-Image definierten Speicherpfade werden verwendet.
`RESTORE DATABASE TEST4 ON /pfad1, /pfad2, /pfad3`	`/pfad1`	Alle Speichergruppen verwenden `/pfad1`, `/pfad2`, `/pfad3` für die zugehörigen Speicherpfade.	Der Datenbankpfad der vorhandenen Datenbank wird verwendet.	Alle Speichergruppen verwenden `/pfad1`, `/pfad2`, `/pfad3` für die zugehörigen Speicherpfade.
`RESTORE DATABASE TEST5 ON E:\neupfad1, F:\neupfad2 DBPATH ON D:`	`D:`	Alle Speichergruppen verwenden `E:\neupfad1`, `F:\neupfad2` für die zugehörigen Speicherpfade.	Der Datenbankpfad der vorhandenen Datenbank wird verwendet.	Alle Speichergruppen verwenden `E:\neupfad1`, `F:\neupfad2` für die zugehörigen Speicherpfade.

In den Fällen, in denen Speicherpfade als Teil der Restoreoperation erneut definiert wurden, werden die Tabellenbereiche, für die die Verwendung von dynamischem Speicher definiert wurde, automatisch an die neuen Pfade umgeleitet. Es ist jedoch nicht möglich, Container, die Tabellenbereichen mit dynamischem Speicher zugeordnet sind, mit dem Befehl **SET TABLESPACE CONTAINERS** explizit umzuleiten. Eine solche Aktion ist nicht zulässig.

Verwenden Sie die Option **-s** des Befehls **db2ckbkp**, um zu prüfen, ob Speichergruppen für eine Datenbank in einem Backup-Image vorhanden sind. Die Speichergruppen und die zugehörigen Speicherpfade werden angezeigt.

Für Datenbanken mit mehreren Datenbankpartitionen hat der Befehl **RESTORE DATABASE** einige zusätzliche Auswirkungen:

1. Die Datenbank muss die gleiche Gruppe von Speicherpfaden in allen Datenbankpartitionen verwenden.
2. Die Ausführung eines Befehls **RESTORE** mit neuen Speicherpfaden ist nur in der Katalog-
   datenbankpartition möglich. Sie versetzt die Datenbank in den Status 'Restore anstehend' (RESTORE_PENDING) in allen anderen Datenbankpartitionen (d. h. Nichtkatalogpartitionen).

Tabelle 11. Auswirkungen eines Befehls RESTORE für Mehrpartitionsdatenbanken

Befehl **RESTORE DATABASE**	Datenbank- partition der Ausführung	Auf der Platte ist keine Datenbank mit demselben Namen vorhanden		Auf der Platte ist eine Datenbank mit demselben Namen vorhanden (gilt auch für Entwurfsdatenbanken)	
		Ergebnis in anderen Datenbank- partitionen	Speicherpfade	Ergebnis in anderen Datenbank- partitionen	Speicherpfade
`RESTORE DATABASE TEST1`	Katalog- datenbank- partition	Eine Entwurfsdatenbank wird mit den Speicherpfaden aus dem Backup-Image in der Katalog-datenbankpartition erstellt. Alle anderen Datenbankpartitionen werden in den Status 'Restore anstehend' (RESTORE_PENDING) versetzt.	Die im Backup-Image definierten Speicherpfade werden verwendet.	Keines. Speicherpfade wurden nicht geändert, sodass es zu keinen Auswirkungen auf andere Datenbankpartitionen kommt.	Die im Backup-Image definierten Speicherpfade werden verwendet.
	Nichtkatalog-datenbank- partition	SQL2542N oder SQL2551N wird zurückgegeben. Wenn keine Datenbank vorhanden ist, muss die Katalogdatenbank-partition zuerst mit Restore wiederhergestellt werden.	N/V	Keines. Speicherpfade wurden nicht geändert, sodass es zu keinen Auswirkungen auf andere Datenbankpartitionen kommt.	Die im Backup-Image definierten Speicherpfade werden verwendet.
`RESTORE DATABASE TEST2 ON /pfad1, /pfad2, /pfad3`	Katalog- datenbank- partition	Eine Entwurfsdatenbank wird mit den im **RESTORE**-Befehl angegebenen Speicherpfaden erstellt. Alle anderen Datenbankpartitionen werden in den Status 'Restore anstehend' (RESTORE_PENDING) versetzt.	Alle Speichergruppen verwenden `/pfad1`, `/pfad2`, `/pfad3` für die zugehörigen Speicherpfade.		Alle Speichergruppen verwenden `/pfad1`, `/pfad2`, `/pfad3` für die zugehörigen Speicherpfade.
	Nichtkatalog-datenbank-partition	SQL1174N wird zurückgegeben. Wenn keine Datenbank vorhanden ist, muss die Katalog-datenbankpartition zuerst mit Restore wiederhergestellt werden. Speicherpfade können im **RESTORE**-Befehl einer Nichtkatalogdatenbank-partition nicht angegeben werden.	N/V	SQL1172N wird zurückgegeben. Neue Speicherpfade können im Befehl **RESTORE** einer Nichtkatalogdatenbankpartition nicht angegeben werden.	N/V

Katalogisieren von Datenbanken

Wenn Sie eine Datenbank erstellen, wird sie automatisch in der Datei für das Systemdatenbankverzeichnis katalogisiert. Sie können außerdem den Befehl **CATALOG DATABASE** verwenden, um eine Datenbank explizit in der Datei für das Systemdatenbankverzeichnis zu katalogisieren.

Informationen zu diesem Vorgang

Mit dem Befehl **CATALOG DATABASE** können Sie eine Datenbank mit einem anderen Aliasnamen oder auch einen Datenbankeintrag katalogisieren, der zuvor mit dem Befehl **UNCATALOG DATABASE** gelöscht wurde.

Obwohl Datenbanken bei ihrer Erstellung automatisch katalogisiert werden, kann es erforderlich werden, eine Datenbank zu katalogisieren. Wenn dieser Fall eintritt, muss die Datenbank vorhanden sein.

Standardmäßig werden Verzeichnisdateien (einschließlich des Datenbankverzeichnisses) im Hauptspeicher zwischengespeichert. Hierzu wird der Konfigurationsparameter für die Verzeichniscacheunterstützung (**dir_cache**) verwendet. Wenn der Verzeichniscache aktiviert ist, wird eine Änderung an einem Verzeichnis (z. B. durch den Befehl **CATALOG DATABASE** oder **UNCATALOG DATABASE**) durch eine andere Anwendung möglicherweise erst wirksam, wenn Ihre Anwendung erneut gestartet wird. Um den Verzeichniscache, der von einer Sitzung des Befehlszeilenprozessors verwendet wird, zu aktualisieren, führen Sie den Befehl **TERMINATE** aus.

In einer partitionierten Datenbank wird in jeder Datenbankpartition ein Cache für Verzeichnisdateien erstellt.

Neben dem Cache auf Anwendungsebene wird auch ein Cache auf Datenbankmanagerebene für interne Suchfunktionen des Datenbankmanagers verwendet. Um diesen „gemeinsamen" Cache zu aktualisieren, führen Sie die Befehle **db2stop** und **db2start** aus.

Vorgehensweise

- Verwenden Sie den Befehl **CATALOG DATABASE** über den Befehlszeilenprozessor, um eine Datenbank unter einem anderen Aliasnamen zu katalogisieren.

  Zum Beispiel wird durch den folgenden Befehl des Befehlszeilenprozessors die Datenbank PERSON1 als HUMANRES katalogisiert:

  ```
  CATALOG DATABASE personl AS humanres
    WITH "Human Resources Database"
  ```

  In diesem Beispiel enthält der Datenbankverzeichniseintrag des Systems den Aliasnamen HUMANRES für die Datenbank, der sich vom Datenbanknamen (PERSON1) unterscheidet.
- Zum Katalogisieren einer Datenbank im Systemdatenbankverzeichnis aus einer Clientanwendung heraus rufen Sie die API sqlecadb auf.
- Verwenden Sie den Befehl **CATALOG DATABASE** über den Befehlszeilenprozessor, um eine Datenbank in einer anderen Instanz als der Standardinstanz zu katalogisieren.

  Im folgenden Beispiel werden Verbindungen zur Datenbank B nun zur Instanz INSTNC_C hergestellt. Die Instanz instnc_c muss bereits als lokaler Knoten katalogisiert sein, bevor dieser Befehl ausgeführt wird.

  ```
  CATALOG DATABASE b as b_on_ic AT NODE instnc_c
  ```

Anmerkung: Der Befehl **CATALOG DATABASE** wird auch auf Clientknoten verwendet, um Datenbanken zu katalogisieren, die sich auf Datenbankservercomputern befinden.

Binden von Dienstprogrammen an die Datenbank

Bei der Erstellung einer Datenbank versucht der Datenbankmanager, die in den Dateien `db2ubind.lst` und `db2cli.lst` aufgeführten Dienstprogramme an die Datenbank zu binden. Diese Dateien befinden sich im Unterverzeichnis `bnd` im entsprechenden Verzeichnis `sqllib`.

Informationen zu diesem Vorgang

Durch das Binden (BIND) eines Dienstprogramms wird ein *Paket* (d. h., ein Objekt) erstellt, das alle Informationen enthält, die zur Verarbeitung bestimmter SQL- und XQuery-Anweisungen aus einer einzigen Quellendatei benötigt werden.

Anmerkung: Wenn diese Dienstprogramme von einem Client ausgeführt werden sollen, müssen sie explizit gebunden werden. Sie müssen sich in dem Verzeichnis befinden, in dem diese Dateien gespeichert sind, um die Pakete in der Datenbank 'sample' zu erstellen. Die Bindedateien befinden sich im Unterverzeichnis `bnd` des Verzeichnisses `sqllib`. Sie müssen auch die Datei `db2schema.bnd` binden, wenn Sie die Datenbank erstellen oder ein Upgrade der Datenbank von einem Client durchführen. Detaillierte Informationen finden Sie im Abschnitt „DB2-CLI-Bindedateien und Paketnamen".

Vorgehensweise

Rufen Sie über die Befehlszeile die folgenden Befehle auf, um Dienstprogramme an eine Datenbank zu binden bzw. neu zu binden:

```
connect to sample
bind @db2ubind.lst
```

Hierbei steht *sample* für den Namen der Datenbank.

Erstellen von Aliasnamen der Datenbank

Ein *Aliasname* ist eine indirekte Methode, auf eine Tabelle, einen Kurznamen oder eine Sicht zu verweisen, sodass eine SQL- oder XQuery-Anweisung vom qualifizierten Namen dieser Tabelle oder Sicht unabhängig sein kann.

Informationen zu diesem Vorgang

Es muss lediglich die Definition des Aliasnamens geändert werden, wenn die Tabelle oder Sicht umbenannt wird. Ein Aliasname kann für andere Aliasnamen erstellt werden. Ein Aliasname kann in einer Sicht- oder Triggerdefinition und in jeder SQL- oder XQuery-Anweisung außer in Definitionen von Prüfungen auf Integritätsbedingungen in Tabellen (Table Check Constraints) verwendet werden, in denen ein vorhandener Tabellen- oder Sichtname angegeben werden kann.

Ein Aliasname kann für eine Tabelle, eine Sicht oder einen Aliasnamen definiert werden, die bzw. der zum Zeitpunkt der Definition noch nicht existiert. Wenn die SQL- oder XQuery-Anweisung, die diesen Aliasnamen enthält, kompiliert wird, muss die Tabelle, Sicht oder der Aliasname vorhanden sein.

Ein Aliasname kann überall dort verwendet werden, wo ein vorhandener Tabellenname verwendet werden kann, und kann auf einen anderen Aliasnamen verweisen, sofern keine rückbezüglichen oder sich wiederholenden Verweise in der Kette der Aliasnamen auftreten.

Als Aliasname kann kein bereits vorhandener Tabellen-, Sicht- oder Aliasname angegeben werden, und der Name darf sich nur auf eine Tabelle in derselben Datenbank beziehen. Der Name einer Tabelle oder Sicht, der in einer Anweisung CREATE TABLE oder CREATE VIEW verwendet wird, darf nicht mit einem Aliasnamen im selben Schema übereinstimmen.

Sie benötigen keine besondere Berechtigung zur Erstellung eines Aliasnamens, sofern der Aliasname sich nicht in einem anderen Schema als dem befindet, das Ihrer aktuellen Berechtigungs-ID eigen ist. In diesem Fall benötigen Sie die Berechtigung DBADM.

Wenn ein Aliasname oder das Objekt, auf das ein Aliasname verweist, gelöscht wird, werden alle von dem Aliasnamen abhängigen Pakete als nicht gültig und alle von dem Aliasnamen abhängigen Sichten und Trigger als funktionsunfähig markiert.

Anmerkung: DB2 for z/OS implementiert zwei verschiedene Konzepte von Aliasnamen: ALIAS und SYNONYM. Diese beiden Konzepte weichen auf folgende Weise von der DB2 Database for Linux, UNIX and Windows ab:

- ALIASnamen in DB2 for z/OS:
  - Sie machen eine spezielle Berechtigung oder ein bestimmtes Zugriffsrecht für ihre Ersteller erforderlich.
  - Sie können nicht auf andere Aliasnamen verweisen.
- SYNONYMe in DB2 for z/OS:
  - Sie können nur von ihrem Ersteller verwendet werden.
  - Sie sind immer ohne Qualifikationsmerkmal.
  - Sie werden gelöscht, wenn die Bezugstabelle gelöscht wird.
  - Sie benutzen keinen Namensbereich mit Tabellen oder Sichten gemeinsam.

Vorgehensweise

Geben Sie in der Befehlszeile Folgendes ein, um einen Aliasnamen zu erstellen:

```
CREATE ALIAS aliasname FOR tabellenname
```

Mit der folgenden SQL-Anweisung wird ein Aliasname WORKERS für die Tabelle EMPLOYEE erstellt:

```
CREATE ALIAS WORKERS FOR EMPLOYEE
```

Der Aliasname wird bei der Kompilierung der Anweisung durch den entsprechenden Tabellen- oder Sichtnamen ersetzt. Wenn der Aliasname oder die Aliasnamenskette nicht in einen Tabellen- oder Sichtnamen aufgelöst werden kann, wird ein Fehler zurückgegeben. Wenn zum Beispiel WORKERS ein Aliasname für die Tabelle EMPLOYEE ist, geschieht bei der Kompilierung Folgendes:

```
SELECT * FROM WORKERS
```

wird effektiv zu

```
SELECT * FROM EMPLOYEE
```

Herstellen einer Verbindung zu verteilten relationalen Datenbanken

Verteilte relationale Datenbanken basieren auf formalen Requester-Server-Protokollen und -Funktionen.

Ein *Anwendungsrequester* unterstützt die Anwendungsseite einer Verbindung. Er setzt eine Datenbankanforderung aus der Anwendung in das Format der Kommunikationsprotokolle um, die in einem Netz verteilter Datenbanken verwendet werden können. Diese Anforderungen werden von einem *Datenbankserver* empfangen und verarbeitet, der sich am anderen Endpunkt der Verbindung befindet. Zusammen steuern der Anwendungsrequester und der Datenbankserver die Kommunikation und die standortbedingten Faktoren, sodass die Anwendung praktisch wie bei einem Zugriff auf eine lokale Datenbank arbeiten kann.

Ein Anwendungsprozess muss eine Verbindung zum Anwendungsserver eines Datenbankmanagers herstellen, bevor SQL-Anweisungen ausgeführt werden können, die auf Tabellen oder Sichten verweisen. Die Anweisung CONNECT stellt eine Verbindung zwischen einem Anwendungsprozess und dem zugehörigen Server her.

Es gibt zwei Arten von CONNECT-Anweisungen:

- CONNECT (Typ 1) unterstützt die Semantik mit einer Einzeldatenbank pro Arbeitseinheit (RUOW = Remote Unit of Work, ferne Arbeitseinheit).
- CONNECT (Typ 2) unterstützt die Semantik mit mehreren Datenbanken pro Arbeitseinheit (anwendungsgesteuerte DUOW = Application-Directed Distributed Unit of Work, anwendungsgesteuerte verteilte Arbeitseinheit).

DB2 Call Level Interface (CLI) und das eingebettete SQL unterstützen einen Verbindungsmodus, der als Modus für *gleichzeitig ablaufende Transaktionen* bezeichnet wird und der mehrere Verbindungen ermöglicht, die jeweils als unabhängige Transaktion behandelt werden. Eine Anwendung kann über mehrere gleichzeitig bestehende Verbindungen zur selben Datenbank verfügen.

Der Anwendungsserver kann sich relativ zu der Umgebung, in der der Prozess eingeleitet wird, auf einem lokalen oder fernen System befinden. Ein Anwendungsserver ist auch dann vorhanden, wenn die Umgebung keine verteilten relationalen Datenbanken verwendet. Eine solche Umgebung enthält ein lokales Verzeichnis, in dem die Anwendungsserver, die in einer Anweisung CONNECT angegeben werden können, beschrieben sind.

Der Anwendungsserver führt die gebundene Form einer statischen SQL-Anweisung aus, mit der auf Tabellen oder Sichten verwiesen wird. Die gebundene Anweisung wird einem Paket entnommen, das der Datenbankmanager zuvor durch eine Bindeoperation erstellt hat.

In den meisten Fällen kann eine mit einem Anwendungsserver verbundene Anwendung Anweisungen und Klauseln verwenden, die vom Datenbankmanager des Anwendungsservers unterstützt werden. Dies gilt auch dann, wenn eine Anwendung über den Anwendungsrequester eines Datenbankmanagers ausgeführt wird, der einige dieser Anweisungen und Klauseln *nicht* unterstützt.

Ferne Arbeitseinheit für verteilte relationale Datenbanken

Die *Funktion für ferne Arbeitseinheiten* (RUOW = Remote Unit of Work) ermöglicht die Vorbereitung und Ausführung von SQL-Anweisungen über eine ferne Einheit.

Ein Anwendungsprozess auf Computersystem „A" kann eine Verbindung zu einem Anwendungsserver auf Computersystem „B" herstellen und innerhalb einer oder mehrerer Arbeitseinheiten (UOWs) eine beliebige Anzahl von statischen oder dynamischen SQL-Anweisungen ausführen, die auf Objekte auf Computersystem „B" zugreifen. Nach Beendigung einer Arbeitseinheit auf Computersystem B kann der Anwendungsprozess eine Verbindung zu einem Anwendungsserver auf Computersystem C herstellen, usw.

Die meisten SQL-Anweisungen können remote vorbereitet und ausgeführt werden. Hierbei gelten allerdings die folgenden Einschränkungen:

- Alle Objekte, auf die in einer einzelnen SQL-Anweisung verwiesen wird, müssen vom selben Anwendungsserver verwaltet werden.
- Alle SQL-Anweisungen in einer Arbeitseinheit müssen vom selben Anwendungsserver ausgeführt werden.

Zu einem bestimmten Zeitpunkt befindet sich ein Anwendungsprozess in einem der vier möglichen *Verbindungsstatus*:

- Verbindung möglich und verbunden

  Ein Anwendungsprozess ist mit einem Anwendungsserver verbunden und CONNECT-Anweisungen können ausgeführt werden.

  Wenn die implizite Verbindung möglich ist:

  – Der Anwendungsprozess nimmt diesen Status an, wenn eine Anweisung CONNECT TO oder CONNECT ohne Operanden im Status 'Verbindung möglich und nicht verbunden' erfolgreich ausgeführt wird.
  – Der Anwendungsprozess könnte aus dem Status 'Implizite Verbindung möglich' in diesen Status versetzt werden, wenn eine beliebige SQL-Anweisung mit Ausnahme von CONNECT RESET, DISCONNECT, SET CONNECTION oder RELEASE ausgeführt wird.

  Unabhängig davon, ob die implizite Verbindung möglich ist, wird dieser Status angenommen, wenn Folgendes gilt:

  – Eine Anweisung CONNECT TO wird im Status 'Verbindung möglich und nicht verbunden' erfolgreich ausgeführt.
  – Eine Anweisung COMMIT oder ROLLBACK wird erfolgreich abgesetzt oder im Status 'Verbindung nicht möglich und verbunden' wird ein erzwungener Rollback ausgeführt.
- Verbindung nicht möglich und verbunden

  Ein Anwendungsprozess ist mit einem Anwendungsserver verbunden, eine Anweisung CONNECT TO kann jedoch nicht erfolgreich ausgeführt werden, um die Anwendungsserver zu ändern. Der Anwendungsprozess kann aus dem Status 'Verbindung möglich und verbunden' in diesen Status versetzt werden, wenn eine beliebige SQL-Anweisung mit Ausnahme von CONNECT TO, CONNECT ohne Operand, CONNECT RESET, DISCONNECT, SET CONNECTION, RELEASE, COMMIT oder ROLLBACK ausgeführt wird.
- Verbindung möglich und nicht verbunden

  Ein Anwendungsprozess ist nicht mit einem Anwendungsserver verbunden. CONNECT TO ist die einzige SQL-Anweisung, die ausgeführt werden kann. Andernfalls wird ein Fehler (SQLSTATE 08003) ausgegeben.

  Unabhängig davon, ob eine implizite Verbindung möglich ist, wird der Anwendungsprozess in diesen Status versetzt, wenn bei der Eingabe einer Anweisung CONNECT TO ein Fehler auftritt oder wenn in einer Arbeitseinheit ein Fehler auftritt, der zu einer Verbindungsunterbrechung und einem Rollback führt. Ein Fehler, der darauf zurückzuführen ist, dass der Anwendungsprozess sich nicht

im Status 'Verbindung möglich' befindet, oder dass der Servername nicht im lokalen Verzeichnis aufgeführt ist, verursacht keine Änderung in diesen Status.

Wenn die implizite Verbindung nicht möglich ist, gilt Folgendes:

– Der Anwendungsprozess befindet sich zu Beginn in diesem Status.
– Die Anweisungen CONNECT RESET und DISCONNECT bewirken eine Änderung in diesen Status.

- Implizite Verbindung möglich (wenn die implizite Verbindung verfügbar ist).

  Wenn die implizite Verbindung verfügbar ist, stellt dieser Status den Anfangsstatus eines Anwendungsprozesses dar. Die Anweisung CONNECT RESET bewirkt eine Änderung in diesen Status. Dieser Status wird auch angenommen, wenn eine Anweisung COMMIT oder ROLLBACK eingegeben wird, während sich das System im Status 'Verbindung nicht möglich und verbunden' befindet, und anschließend im Status 'Verbindung möglich und verbunden' eine Anweisung DISCONNECT eingegeben wird.

Die Verfügbarkeit der impliziten Verbindung wird mithilfe von Installationsoptionen, Umgebungsvariablen und Authentifizierungseinstellungen festgelegt.

Die Ausführung aufeinanderfolgender CONNECT-Anweisungen stellt keinen Fehler dar, weil CONNECT selbst den Anwendungsprozess nicht aus dem Status 'Verbindung möglich' in einen anderen Status versetzt. Die Ausführung aufeinanderfolgender CONNECT RESET-Anweisungen stellt jedoch einen Fehler dar. Außerdem ist es auch ein Fehler, eine andere SQL-Anweisung als CONNECT TO, CONNECT RESET, CONNECT ohne Operand, SET CONNECTION, RELEASE, COMMIT oder ROLLBACK und anschließend eine Anweisung CONNECT TO auszuführen. Um diesen Fehler zu vermeiden, muss vor der Anweisung CONNECT TO eine Anweisung CONNECT RESET, DISCONNECT (mit vorausgehender Anweisung COMMIT oder ROLLBACK), COMMIT oder ROLLBACK ausgeführt werden.

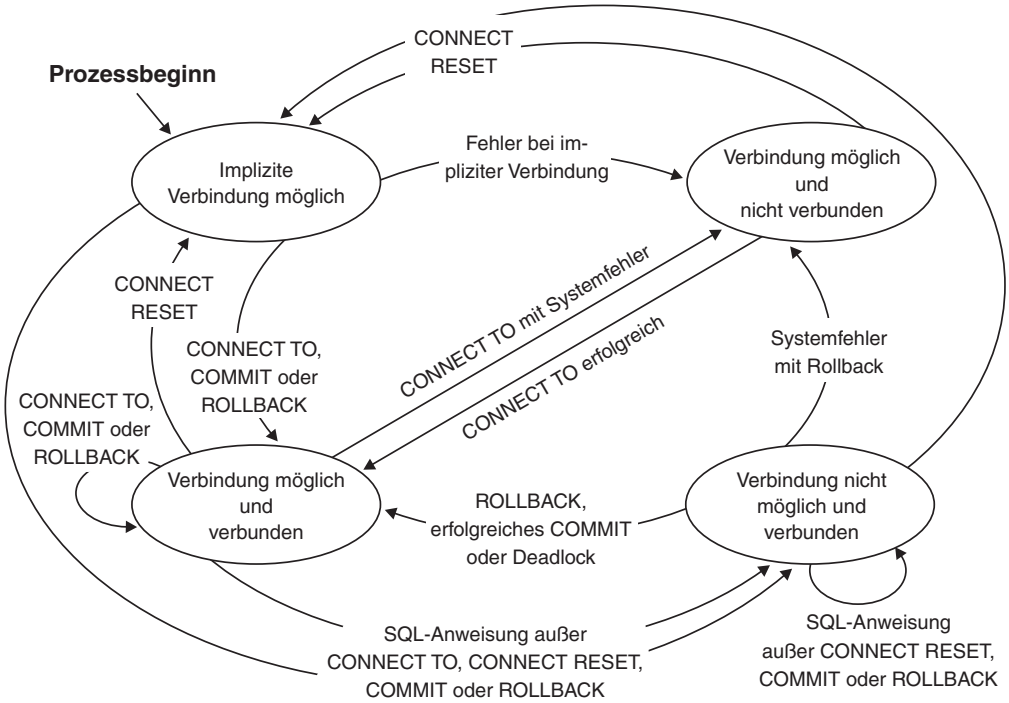


Abbildung 4. Verbindungsstatusänderungen bei Verfügbarkeit einer impliziten Verbindung

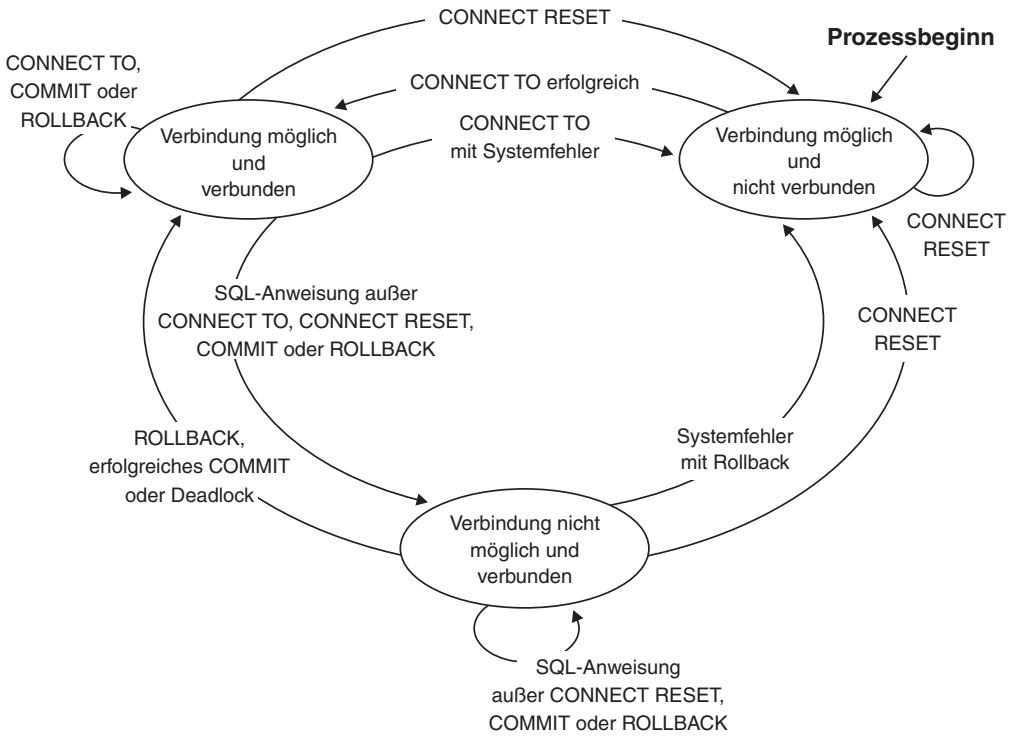


Abbildung 5. Verbindungsstatusänderungen bei Nichtverfügbarkeit einer impliziten Verbindung

Anwendungsgesteuerte verteilte Arbeitseinheit

Die *Funktion für anwendungsgesteuerte DUOWs* (DUOW = Distributed Unit of Work, verteilte Arbeitseinheit) ermöglicht die ferne Vorbereitung und Ausführung von SQL-Anweisungen.

Ein Anwendungsprozess auf dem Computersystem „A" kann eine Verbindung zu einem Anwendungsserver auf Computersystem „B" herstellen, indem er eine Anweisung CONNECT oder SET CONNECTION absetzt. Der Anwendungsprozess kann dann eine beliebige Anzahl statischer und dynamischer SQL-Anweisungen ausführen, die auf Objekte auf dem Computersystem „B" zugreifen, bevor er die UOW beendet. Alle Objekte, auf die in einer einzelnen SQL-Anweisung verwiesen wird, müssen vom selben Anwendungsserver verwaltet werden. Anders als bei der Funktion für ferne Arbeitseinheiten (RUOWs) kann eine beliebige Anzahl von Anwendungsservern an derselben UOW beteiligt werden. Eine Commit- oder Rollbackoperation beendet die UOW.

Eine anwendungsgesteuerte DUOW verwendet eine Verbindung des Typs 2. Eine Verbindung vom *Typ 2* verbindet einen Anwendungsprozess mit dem angegebenen Anwendungsserver und legt die Regeln für anwendungsgesteuerte DUOWs fest.

Für einen Typ 2-Anwendungsprozess gilt Folgendes:

- Der Anwendungsprozess weist immer den Status 'Verbindung möglich' auf.
- Der Anwendungsprozess befindet sich entweder im Status 'Verbunden' oder im Status 'Nicht verbunden'.
- Der Anwendungsprozess verfügt über null oder mehr Verbindungen.

Jede Verbindung eines Anwendungsprozesses wird für die Verbindung durch den Aliasnamen der Datenbank des Anwendungsservers eindeutig identifiziert.

Eine einzelne Verbindung weist immer einen der folgenden Verbindungsstatus auf:

- Aktuell und angehalten
- Aktuell und Freigabe anstehend
- Ruhend und angehalten
- Ruhend und Freigabe anstehend

Ein Anwendungsprozess des Typs 2 befindet sich zu Beginn im Status 'Nicht verbunden' und verfügt nicht über Verbindungen. Eine Verbindung befindet sich zu Beginn im Status 'Aktuell und angehalten'.

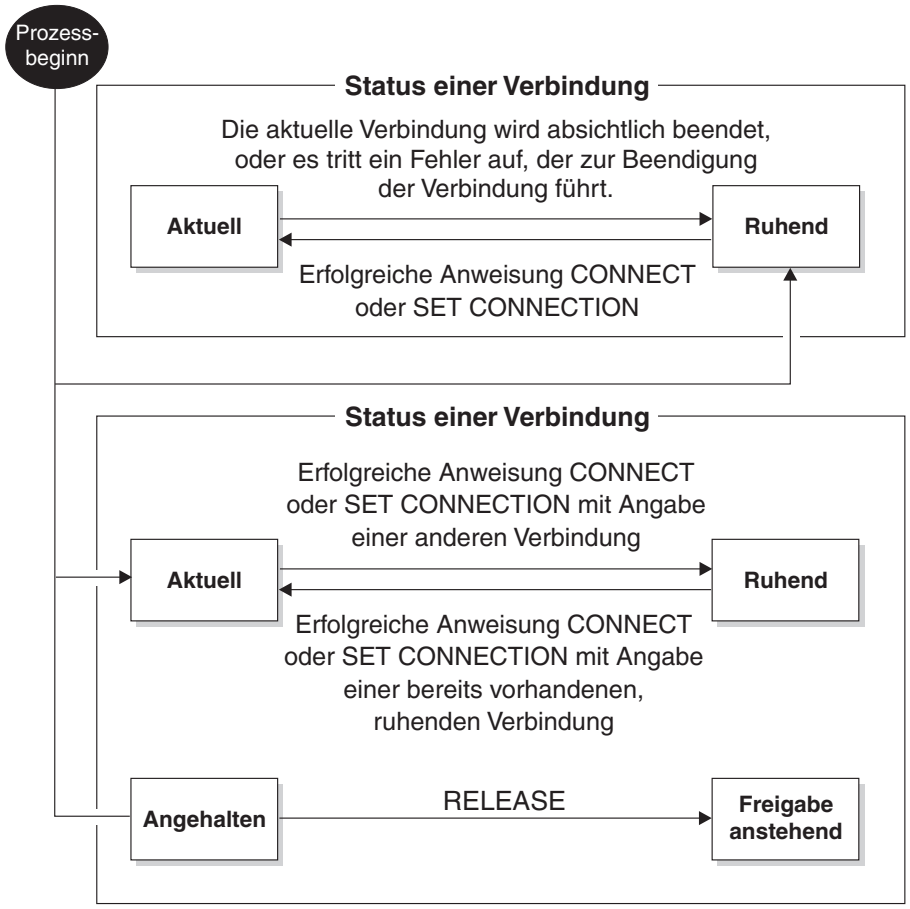


Abbildung 6. Verbindungsstatusänderungen bei anwendungsgesteuerten verteilten DUOWs

Verbindungsstatus von Anwendungsprozessen

Für die Ausführung einer Anweisung CONNECT gelten bestimmte Regeln.

Diese Regeln lauten wie folgt:

- Ein Kontext kann nicht gleichzeitig über mehrere Verbindungen zum selben Anwendungsserver verfügen.
- Wenn ein Anwendungsprozess eine Anweisung SET CONNECTION ausführt, muss als Positionsname eine bereits vorhandene Verbindung aus der Gruppe der Verbindungen für den Anwendungsprozess angegeben werden.

- Wenn ein Anwendungsprozess eine Anweisung CONNECT ausführt und die Option SQLRULES(STD) aktiviert wurde, darf der angegebene Servername *nicht* mit der Angabe für eine bereits vorhandene Verbindung aus der Gruppe der Verbindungen für den Anwendungsprozess übereinstimmen. Eine Beschreibung der Option SQLRULES finden Sie unter „Optionen zur Regelung der Semantik von Arbeitseinheiten" auf Seite 147.

Wenn ein Anwendungsprozess über eine aktuelle Verbindung verfügt, befindet sich der Anwendungsprozess im Status *Verbunden*. Das Sonderregister CURRENT SERVER enthält den Namen des Anwendungsservers für die aktuelle Verbindung. Der Anwendungsprozess kann SQL-Anweisungen ausführen, die auf Objekte verweisen, die von dem Anwendungsserver verwaltet werden.

Ein Anwendungsprozess, der sich im Status 'Nicht verbunden' befindet, wird in den Status 'Verbunden' überführt, wenn eine Anweisung CONNECT oder SET CONNECTION erfolgreich ausgeführt werden konnte. Wenn keine Verbindung vorhanden ist, jedoch SQL-Anweisungen ausgegeben werden, wird eine implizite Verbindung hergestellt. Dies gilt allerdings nur, wenn in der Umgebungsvariablen **DB2DBDFT** der Name einer Standarddatenbank angegeben wurde.

Wenn ein Anwendungsprozess über keine aktuelle Verbindung verfügt, befindet sich der Anwendungsprozess im Status *Nicht verbunden*. Als einzige SQL-Anweisungen können die Anweisungen CONNECT, DISCONNECT ALL, DISCONNECT (Angabe einer Datenbank), SET CONNECTION, RELEASE, COMMIT, ROLLBACK sowie lokale SET-Anweisungen ausgeführt werden.

Ein Anwendungsprozess im Status *Verbunden* wird in den Status *Nicht verbunden* überführt, wenn die momentan vorhandene, aktuelle Verbindung absichtlich beendet wird oder wenn eine SQL-Anweisung fehlschlägt und zu einer Rollbackoperation auf dem Anwendungsserver und dem Verlust der Verbindung führt. Verbindungen werden nach der erfolgreichen Ausführung einer Anweisung DISCONNECT oder nach der Ausführung einer Anweisung COMMIT absichtlich beendet, wenn die Verbindung sich im Status 'Freigabe anstehend' befindet. (Wenn die Precompileroption DISCONNECT auf den Wert AUTOMATIC eingestellt ist, werden alle Verbindungen beendet. Wenn sie auf den Wert CONDITIONAL eingestellt ist, werden hingegen alle Verbindungen beendet, die nicht über geöffnete WITH HOLD-Cursor verfügen.)

Verbindungsstatus

Es gibt zwei Typen von Verbindungsstatus: Die Status „Angehalten und Freigabe anstehend" und die Status „Aktuell und Ruhend".

Wenn ein Anwendungsprozess eine Anweisung CONNECT ausführt und der Servername für den Anwendungsrequester definiert wurde, sich jedoch nicht in der Gruppe der vorhandenen Verbindungen für den Anwendungsprozess befindet, gilt Folgendes: (i) Die aktuelle Verbindung wird in den Verbindungsstatus *Ruhend* versetzt, der Servername wird zur Gruppe der Verbindungen hinzugefügt und die neue Verbindung wird sowohl in den Verbindungsstatus *Aktuell* als auch in den Verbindungsstatus *Angehalten* versetzt.

Wenn sich der Servername bereits in der Gruppe der vorhandenen Verbindungen für den Anwendungsprozess befindet und die Anwendung mit der Option SQLRULES(STD) vorkompiliert wurde, wird ein Fehler (SQLSTATE 08002) ausgegeben.

Status 'Angehalten' und 'Freigabe anstehend'. Mit der Anweisung RELEASE wird gesteuert, ob sich eine Verbindung im Status 'Angehalten' oder 'Freigabe anstehend' befindet. Der Status *Freigabe anstehend* bedeutet, dass nach der nächsten erfolgreichen Ausführung einer Commitoperation eine Verbindungsunterbrechung durchgeführt wird. (Eine Rollbackoperation hat keine Auswirkungen auf Verbindungen.) Der Status *Angehalten* bedeutet, dass nach der nächsten Ausführung einer Commitoperation *keine* Verbindungsunterbrechung durchgeführt wird.

Alle Verbindungen befinden sich zu Beginn im Status 'Angehalten' und können mit der Anweisung RELEASE in den Status 'Freigabe anstehend' versetzt werden. Nachdem sie sich im Status 'Freigabe anstehend' befindet, kann eine Verbindung nicht wieder zurück in den Status 'Angehalten' versetzt werden. Eine Verbindung bleibt über Grenzen von UOWs (Arbeitseinheiten) hinweg im Status 'Freigabe anstehend', wenn eine Anweisung ROLLBACK aufgeführt wird oder wenn eine fehlgeschlagene Commitoperation zu einer ROLLBACK-Operation führt.

Selbst wenn eine Verbindung nicht explizit für die Freigabe markiert ist, kann sie trotzdem aufgrund einer Commitoperation unterbrochen werden, wenn diese die Bedingungen der Precompileroption DISCONNECT erfüllt.

Status 'Aktuell' und 'Ruhend'. Unabhängig davon, ob sich eine Verbindung im Status 'Angehalten' oder im Status 'Freigabe anstehend' befindet, kann sie sich außerdem im Status 'Aktuell' oder im Status 'Ruhend' befinden. Eine Verbindung im Status *Aktuell* wird zur Ausführung von SQL-Anweisungen verwendet, solange sich dieser Status nicht ändert. Eine Verbindung im Status *Ruhend* ist eine Verbindung, die momentan nicht aktuell ist.

Die einzigen SQL-Anweisungen, die über eine ruhende Verbindung ausgeführt werden können, sind COMMIT, ROLLBACK, DISCONNECT und RELEASE. Die Anweisungen SET CONNECTION und CONNECT ändern den Verbindungsstatus des angegebenen Servers in 'Aktuell'. Alle vorhandenen Verbindungen werden in den Status 'Ruhend' versetzt oder bleiben weiterhin in diesem Status. Zu einem bestimmten Zeitpunkt kann sich immer nur eine Verbindung im Status 'Aktuell' befinden. Wenn eine ruhende Verbindung innerhalb derselben UOW aktiviert wird, dann entspricht der Status aller Sperren, Cursor und vorbereiteten Anweisungen dem Status, in dem sich diese befanden, als die Verbindung das letzte Mal den Status 'Aktuell' hatte.

Beendigung einer Verbindung

Wenn eine Verbindung beendet wird, werden alle Ressourcen, die vom Anwendungsprozess über die Verbindung belegt wurden, und alle Ressourcen, die zum Herstellen und Verwalten der Verbindung verwendet wurden, freigegeben. Wenn beispielsweise der Anwendungsprozess eine Anweisung RELEASE ausführt, werden alle geöffneten Cursor geschlossen, wenn die Verbindung während der nächsten Commitoperation beendet wird.

Eine Verbindung kann auch aufgrund eines Kommunikationsfehlers beendet werden. Wenn sich diese Verbindung im Status 'Aktuell' befindet, dann wird der Anwendungsprozess in den Status 'Nicht verbunden' versetzt.

Alle Verbindungen für einen Anwendungsprozess werden beendet, wenn der Prozess beendet wird.

Anpassen einer Anwendungsumgebung mithilfe der Verbindungsprozedur

Die Verbindungsprozedur bietet eine Möglichkeit, mit der es Anwendungen in Ihrer Umgebung ermöglicht wird, eine bestimmte Prozedur direkt beim Herstellen einer Verbindung implizit auszuführen. Mit dieser Prozedur können Sie eine Anwendungsumgebung von einem zentralen Steuerungspunkt aus an eine Datenbank anpassen.

Beispielsweise können Sie in der Verbindungsprozedur Sonderregister, wie CURRENT_PATH, auf nicht standardmäßige Werte setzen, indem Sie die Anweisung SET CURRENT PATH aufrufen. Dieser neue Wert für CURRENT_PATH dient nun als wirksamer Standardwert für CURRENT_PATH für alle Anwendungen.

Jede in der Datenbank erstellte Prozedur, die den Benennungs- und Parametereinschränkungen entspricht, kann als Verbindungsprozedur für die betreffende Datenbank verwendet werden. Die Anpassungslogik wird in Form einer Prozedur bereitgestellt, die in derselben Datenbank erstellt wurde und die alle normalerweise von einer Prozedur ausgeführten Aktionen vornehmen darf, wie zum Beispiel das Absetzen von SQL-Anweisungen.

Mit dem Datenbankkonfigurationsparameter **connect_proc** wird die Verbindungsprozedur angegeben, die für alle Verbindungen zur Datenbank verwendet wird. Aktualisieren Sie den Parameter **connect_proc**, um den Namen der Verbindungsprozedur festzulegen und sie zu aktivieren. Eine Datenbankverbindung ist erforderlich, um einen Wert des Parameters **connect_proc** ungleich null zu aktualisieren. Nach der Definition des Parameters **connect_proc** ist es erforderlich, dass die Sitzungsberechtigungs-ID jeder neuen Verbindung über das Zugriffsrecht EXECUTE für die angegebene Verbindungsprozedur verfügt; die Sitzungsberechtigungs-ID kann dieses Zugriffsrecht direkt oder indirekt über eine der zugehörigen Gruppen, Rollen oder PUBLIC besitzen.

Die Verbindungsprozedur wird zum Abschluss einer erfolgreichen Verbindungsverarbeitung und vor der Verarbeitung nachfolgender Anforderungen für dieselbe Verbindung implizit auf dem Server ausgeführt. Nach der erfolgreichen Ausführung der Verbindungsprozedur schreibt der Datenbankmanager alle Änderungen fest, die von der Verbindungsprozedur vorgenommen wurden. Schlägt die Verbindungsprozedur fehl, werden alle von ihr vorgenommenen Änderungen rückgängig gemacht und der Verbindungsversuch schlägt mit einer Fehlernachricht fehl.

Anmerkung: Alle in der Verbindungsprozedur an einem Sonderregister vorgenommenen Änderungen werden in der daraus resultierenden Sitzung reflektiert, auch nach Abschluss der Prozedur.

Wichtig: Alle von der Verbindungsprozedur zurückgegebenen Fehler bewirken einen Fehler beim Versuch des Verbindungsaufbaus. Der von der Ausführung der Verbindungsprozedur zurückgegebene Fehler wird an die Anwendung gesendet. Wenn Sie die Verbindungsprozedur ändern und den Fehler beheben wollen, müssen Sie die Definition des Parameters **connect_proc** aufheben, um erfolgreiche Verbindungen zu ermöglichen, bis der Fehler behoben ist.

Empfehlungen für die Verbindungsprozedur

Um Probleme mit der Verbindungsprozedur zu vermeiden, müssen Sie sicherstellen, dass Ihre Verbindungsprozedur den folgenden Empfehlungen entspricht:

- Die Logik der Verbindungsprozedur sollte einfach sein.

Die Verwendung einer Verbindungsprozedur hat Auswirkungen auf die Leistung von CONNECT-Befehlen bei jeder Verbindungsherstellung, da zusätzliche Verarbeitungsschritte ausgeführt werden. Die Auswirkungen auf die Leistung können erheblich sein, wenn die Prozedur nicht effizient ist oder Verzögerungen wie beispielsweise Zugriffskonflikte bei Sperren auftreten.

Stellen Sie sicher, dass die Prozedur umfassend getestet wird, bevor sie als Verbindungsprozedur verwendet wird.

- Vermeiden Sie es, in der Verbindungsprozedur auf Objekte zuzugreifen, die gelöscht oder geändert werden sollen.

  Wenn ein abhängiges Objekt in der Verbindungsprozedur gelöscht wird oder wenn Zugriffsrechte für abhängige Objekte widerrufen werden, schlägt die Verbindungsprozedur unter Umständen fehl. Ein von der Prozedur zurückgegebener Fehler kann die Herstellung von neuen Verbindungen zur Datenbank auf der Basis der Logik Ihrer Prozedur blockieren.
- Vermeiden Sie das Aufrufen einer anderen Prozedur in der Verbindungsprozedur.

  Prozeduren, die von der Verbindungsprozedur aufgerufen werden, können im Gegensatz zur Verbindungsprozedur selbst gelöscht oder geändert werden. Wenn Prozeduren, die von der Verbindungsprozedur aufgerufen werden, inaktiviert oder gelöscht werden, schlägt die Verbindungsprozedur unter Umständen fehl. Ein von der Verbindungsprozedur zurückgegebener Fehler kann die Herstellung von neuen Verbindungen zur Datenbank auf der Basis der Logik Ihrer Prozedur blockieren. Beachten Sie darüber hinaus, dass Sonderregister, die in Prozeduren geändert werden, die von der Verbindungsprozedur aufgerufen werden, keine Änderung der Sonderregister in der aufrufenden Umgebung bewirken (im Gegensatz zu den Sonderregistern, die in der Verbindungsprozedur selbst geändert werden und sich auf die Anwendung auswirken).
- Vermeiden Sie die Angabe der Klausel COMMIT ON RETURN in der Verbindungsprozedur.

  Ein internes Commit wird nach dem impliziten Aufruf der Verbindungsprozedur verarbeitet. Wenn die Klausel COMMIT ON RETURN YES angegeben wird, verarbeitet der Datenbankmanager mehrere Commitaufrufe, was sich auf die Leistung auswirken kann. Die Angabe von COMMIT ON RETURN NO hat keine Auswirkungen auf die Verbindungsprozedurverarbeitung.
- Geben Sie alle Ressourcen frei und schließen Sie alle Cursor, bevor Sie die Verbindungsprozedur beenden.

  Anwendungen können auf Ressourcen, die von der Verbindungsprozedur offen gelassen wurden (wie WITH HOLD-Cursor), nicht zugreifen. Die Ressourcen, die von der Verbindungsprozedur nach der Commitverarbeitung gehalten werden, können erst nach dem Beenden der Anwendung freigegeben werden.
- Erteilen Sie PUBLIC das Zugriffsrecht EXECUTE für die Verbindungsprozedur.

  Eine Verbindungsprozedur ist nicht zur Steuerung des Datenbankzugriffs konzipiert. Die Zugriffssteuerung erfolgt durch das Erteilen von Datenbankberechtigungen an Benutzer.
- Vermeiden Sie die Verwendung unterschiedlicher Werte für den Parameter **connect_proc** für verschiedene Datenbankpartitionen.

  Die Verwendung verschiedener Verbindungsprozeduren für unterschiedliche Datenbankpartitionen kann zu inkonsistentem Anwendungsverhalten führen, das jeweils von der Datenbankpartition abhängt, zu der Benutzer eine Verbindung hergestellt haben. Darüber hinaus wird die Datenbankumgebung dadurch auch komplexer und problematischer in der Verwaltung.

Hinweise zur Verwendung der Verbindungsprozedur

Für die Verbindungsprozedur gelten die folgenden Einschränkungen:

- Solange der Parameter **connect_proc** gesetzt ist, kann keine Prozedur mit demselben Namen wie eine vorhandene Verbindungsprozedur erstellt werden.
- Nur eine Prozedur mit genau null Parametern kann als Verbindungsprozedur verwendet werden. Wenn der Parameter **connect_proc** gesetzt ist, darf keine andere Prozedur mit demselben zweiteiligen Namen in der Datenbank vorhanden sein.
- Der Verbindungsprozedurname (sowohl Schemaname als auch Prozedurname) darf nur die folgenden Zeichen enthalten:
  - A - Z
  - a - z
  - _ (Unterstreichungszeichen)
  - 0 - 9

  Darüber hinaus müssen der Schemaname und der Prozedurname den Regeln für einen Standardbezeichner entsprechen.
- Während der Parameter **connect_proc** gesetzt ist, kann die Verbindungsprozedur nicht gelöscht oder geändert werden.

  Wenn Sie die Verbindungsprozedur ändern oder löschen möchten, ändern Sie den Wert des Parameters **connect_proc** in null oder in den Namen einer anderen Prozedur.
- Eine Verbindungsprozedur kann Clientinformationsfelder, die von der API sqleseti oder den SQLSetConnectAttr-CLI-Funktionen definiert werden, nicht verwenden.

  Das Sonderregister für diese Felder enthält deren Standardserverwerte vor der Ausführung der Verbindungsprozedur. Die Clientinformationsfelder bzw. das SQL-Sonderregister, die/das durch das Aufrufen der API sqleseti, der CLI-Funktion SQLSetConnectAttr oder der CLI-Funktion SQLSetEnvAttr definiert werden/wird (z. B. CLIENT USERID, CLIENT ACCTNG, CLIENT APPLNAME und CLIENT WRKSTNNAME), sind/ist beim Ausführen der Verbindungsprozedur noch nicht aktualisiert.
- Die Clientinformationsfelder bzw. das SQL-Sonderregister, die/das durch das Aufrufen der API sqleseti, der CLI-Funktion SQLSetConnectAttr oder der CLI-Funktion SQLSetEnvAttr definiert werden/wird, und der Methodensatz ClientAccountingInformation von IBM Data Server Driver for JDBC and SQLJ haben Priorität gegenüber den in der Verbindungsprozedur definierten Sonderregisterwerten und überschreiben diese.
- Nur Sonderregister, die direkt durch die Verbindungsprozedur definiert werden, bleiben nach der Rückgabe von der Verbindungsprozedur weiterhin definiert. Die Aufrufe für verschachtelte Routinen in der Verbindungsprozedur ändern die Einstellungen der Sonderregister in der aufrufenden Umgebung nicht.

Beispiele für die Implementierung der Verbindungsprozedur

Die folgenden Beispiele zeigen einige Muster für die Verbindungsprozedur und deren Aktivierung in der Datenbank:

Beispiel 1

1. Definieren Sie die SQL-Prozedur NEWTON.CONNECTPROC zur Definition eines Sonderregisters auf der Basis von SESSION_USER.

```
CREATE PROCEDURE NEWTON.CONNECTPROC ( )
READS SQL DATA
LANGUAGE SQL
   BEGIN

   --Sonderregister auf der Basis der Sitzungsbenutzer-ID definieren
      CASE SESSION_USER
        WHEN 'USERA' THEN
         SET CURRENT LOCALE LC_TIME 'fr_FR';

        WHEN 'USERB' THEN
         SET CURRENT LOCALE LC_TIME 'de_DE';
   ELSE
         SET CURRENT LOCALE LC_TIME 'au_AU';

   END CASE;

   END %
```

Mit dieser Prozedur werden Einstellungen für das Sonderregister CURRENT LOCALE LC_TIME festgelegt, wobei Sonderfallwerte für die Benutzer USERA und USERB definiert werden.

2. Erteilen Sie der Gruppe PUBLIC das Zugriffsrecht EXECUTE für die Verbindungsprozedur:

   ```
   GRANT EXECUTE ON PROCEDURE NEWTON.CONNECTPROC TO PUBLIC
   ```

   .

3. Aktualisieren Sie den Parameter **connect_proc** so, dass diese neue Prozedur für jede neue Verbindung aufgerufen wird:

   ```
   db2 update db cfg using connect_proc "NEWTON.CONNECTPROC"
   ```

   Die Verbindungsprozedur NEWTON.CONNECTPROC wird nun automatisch für alle nachfolgenden CONNECT-Anforderungen für eine neue Verbindung aufgerufen. Das Sonderregister CURRENT LOCALE LC_TIME wird auf der Basis von SESSION USER definiert.

Beispiel 2

1. Definieren Sie eine Prozedur für neue Verbindungen und rufen Sie diese auf, um die Anfangswerte der Sonderregister für diese Verbindungen anzupassen.

   ```
   CREATE PROCEDURE MYSCHEMA.CONNECTPROC
     (   )
     EXTERNAL NAME 'parts!connectproc'
     DBINFO
     READS SQL DATA
     LANGUAGE C
     PARAMETER STYLE SQL
   ```

   Diese Prozedur liest in der Datenbanktabelle MYSCHEMA.CONNECTDEFAULTS, um die Werte zu ermitteln, die in den Sonderregistern CURRENT SCHEMA, CURRENT PATH und CURRENT QUERY OPTIMIZATION auf der Basis der Gruppen, die der Berechtigungs-ID der neuen Verbindung zugeordnet sind, definiert werden sollen. Darüber hinaus wird der Wert der globalen Variable MYSCHEMA.SECURITY_SETTING auf der Basis der Informationen in derselben Tabelle festgelegt.

2. Erteilen Sie der Gruppe PUBLIC das Zugriffsrecht EXECUTE für die Verbindungsprozedur:

   ```
   GRANT EXECUTE ON PROCEDURE MYSCHEMA.CONNECTPROC TO PUBLIC
   ```

   .

3. Aktualisieren Sie den Parameter **connect_proc** so, dass diese neue Prozedur für jede neue Verbindung aufgerufen wird:

   ```
   db2 update db cfg using connect_proc "MYSCHEMA.CONNECTPROC"
   ```

   Die Verbindungsprozedur MYSCHEMA.CONNECTPROC wird nun automatisch für alle nachfolgenden CONNECT-Anforderungen für eine neue Verbindung aufgerufen.

Optionen zur Regelung der Semantik von Arbeitseinheiten

Die Semantik für die Verwaltung von Verbindungen vom Typ 2 wird von einer Reihe von Precompileroptionen bestimmt. Diese Optionen werden in der nachstehenden Liste zusammengefasst. Die jeweiligen Standardwerte sind in Fettdruck hervorgehoben und unterstrichen.

- CONNECT (**1** | 2). Gibt an, ob CONNECT-Anweisungen als Anweisungen des Typs 1 oder Typs 2 verarbeitet werden sollen.
- SQLRULES (**DB2** | STD). Gibt an, ob CONNECT-Anweisungen des Typs 2 auf der Basis der DB2-Regeln verarbeitet werden sollen, die für CONNECT eine Umschaltung zu einer ruhenden Verbindung zulassen, oder ob die Regeln des SQL92-Standards verwendet werden sollen, die diese Vorgehensweise nicht zulassen.
- DISCONNECT (**EXPLICIT** | CONDITIONAL | AUTOMATIC). Gibt an, welche Datenbankverbindungen unterbrochen werden sollen, wenn eine Commitoperation ausgeführt wird:
  - Verbindungen, die mit der SQL-Anweisung RELEASE (EXPLICIT) explizit zur Freigabe markiert wurden.
  - Verbindungen, die nicht über geöffnete WITH HOLD-Cursor verfügen und Verbindungen, die für die Freigabe (CONDITIONAL) markiert sind.
  - Alle Verbindungen (AUTOMATIC).
- SYNCPOINT (**ONEPHASE** | TWOPHASE | NONE). Gibt an, wie COMMIT- und ROLLBACK-Operationen zwischen mehreren Datenbankverbindungen koordiniert werden sollen. Diese Option wird ignoriert und dient nur zur Gewährleistung der Abwärtskompatibilität.
  - Aktualisierungen können in der Arbeitseinheit nur in einer Datenbank durchgeführt werden. Alle anderen Datenbanken sind schreibgeschützt (ONEPHASE). Alle Aktualisierungsversuche in anderen Datenbanken führen zu einem Fehler (SQLSTATE 25000).
  - Zur Laufzeit wird ein Transaktionsmanager (TM) verwendet, um zweiphasige Commitoperationen in den Datenbanken zu koordinieren, die dieses Protokoll (TWOPHASE) unterstützen.
  - Zur Ausführung des zweiphasigen Commits wird kein Transaktionsmanager (TM) verwendet und die Verwendung einer einzelnen Aktualisierungskomponente und mehrerer Eingabekomponenten wird nicht erzwungen (NONE). Wenn eine COMMIT- oder ROLLBACK-Operation ausgeführt wird, werden für alle Datenbanken individuelle COMMIT- oder ROLLBACK-Operationen übergeben. Wenn mindestens eine der ROLLBACK-Operationen fehlschlägt, wird ein Fehler (SQLSTATE 58005) ausgegeben. Wenn mindestens eine COMMIT-Operation fehlschlägt, wird ein weiterer Fehler (SQLSTATE 40003) ausgegeben.

Um während der Laufzeit eine der oben aufgeführten Optionen außer Kraft zu setzen, können Sie den Befehl **SET CLIENT** oder die Anwendungsprogrammierschnittstelle (API) sqlesetc verwenden. Ihre aktuellen Einstellungen können mit dem Befehl **QUERY CLIENT** oder der API sqleqryc abgerufen werden. Beachten Sie hierbei,

dass es sich dabei nicht um SQL-Anweisungen, sondern um APIs handelt, die in verschiedenen Hostprogrammiersprachen und im Befehlszeilenprozessor (CLP) definiert wurden.

Wichtige Hinweise zur Datendarstellung

In verschiedenen Systemen werden Daten auf unterschiedliche Weise dargestellt. Wenn Daten von einem System auf ein anderes verschoben werden, muss in bestimmten Fällen eine Datenkonvertierung durchgeführt werden.

Produkte mit DRDA-Unterstützung führen erforderliche Konvertierungsoperationen auf dem Empfangssystem automatisch durch.

Zum Konvertieren numerischer Daten muss das System über Angaben zum Datentyp und dazu verfügen, wie die Daten vom sendenden System dargestellt werden. Zum Konvertieren von Zeichenfolgen sind weitere Informationen erforderlich. Die Zeichenfolgekonvertierung hängt von der verwendeten Codepage der Daten und von der Operation ab, die für die Daten ausgeführt werden soll. Zeichenkonvertierungen werden auf der Basis von IBM Character Data Representation Architecture (CDRA) durchgeführt. Weitere Informationen zur Zeichenkonvertierung finden Sie im Handbuch *Character Data Representation Architecture: Reference & Registry* (IBM Form SC09-2190-00).

Anzeigen des Inhalts der Datei des lokalen oder des Systemdatenbankverzeichnisses

Mit dem Befehl **LIST DATABASE DIRECTORY** können Sie die Informationen zu den Datenbanken anzeigen, die sich auf Ihrem System befinden.

Vorbereitende Schritte

Sie können den Inhalt der Dateien des lokalen bzw. des Systemdatenbankverzeichnisses erst anzeigen, wenn Sie eine Instanz und eine Datenbank erstellt haben.

Vorgehensweise

- Wenn Sie den Inhalt der Datei des lokalen Datenbankverzeichnisses anzeigen wollen, geben Sie den folgenden Befehl ein:

  ```
  LIST DATABASE DIRECTORY ON position
  ```

  Dabei ist *position* die Position der Datenbank.
- Wenn Sie den Inhalt der Datei des Systemdatenbankverzeichnisses anzeigen wollen, setzen Sie den Befehl **LIST DATABASE DIRECTORY** *ohne* Angabe der Speicherposition der Datei für das Datenbankverzeichnis ab.

Löschen von Datenbanken

Das Löschen einer Datenbank kann weit reichende Auswirkungen haben, da alle Objekte der Datenbank, Container und zugehörige Dateien gelöscht werden. Die gelöschte Datenbank wird aus den Datenbankverzeichnissen entfernt (entkatalogisiert).

Vorgehensweise

- Geben Sie Folgendes ein, um eine Datenbank über die Befehlszeile zu löschen:
  `DROP DATABASE name`

- Rufen Sie zum Löschen einer Datenbank aus einer Clientanwendung heraus die API sqledrpd auf.
- Rufen Sie zum Löschen einer Datenbank auf einem bestimmten Datenbankpartitionsserver die API sqledpan auf.
- Zum Löschen einer Datenbank mithilfe von IBM Data Studio klicken Sie mit der rechten Maustaste auf die Datenbank und wählen Sie den Taskassistenten zum Löschen der Datenbank aus.

Beispiel

Mit dem folgenden Befehl wird die Beispieldatenbank SAMPLE gelöscht:

```
DROP DATABASE SAMPLE
```

Anmerkung: Wenn Sie die Beispieldatenbank SAMPLE löschen, sie jedoch erneut benötigen, können Sie sie erneut erstellen.

Zugehörige Informationen:

Löschen von Aliasnamen

Wenn Sie einen Aliasnamen löschen, wird die zugehörige Beschreibung aus dem Katalog gelöscht; alle Pakete und zwischengespeicherten dynamischen Abfragen, die auf den Aliasnamen verweisen, werden inaktiviert. Alle vom gelöschten Aliasnamen abhängigen Sichten und Trigger werden als funktionsunfähig markiert.

Vorgehensweise

Geben Sie zum Löschen von Aliasnamen die Anweisung DROP über die Befehlszeile ein:

```
DROP ALIAS mitarbeitername
```

Kapitel 6. Datenbankpartitionen

Eine *Datenbankpartition* ist ein Teil einer Datenbank, der aus eigenen Daten, Indizes, Konfigurationsdateien und Transaktionsprotokollen besteht. Eine Datenbankpartition wird manchmal auch als Knoten oder Datenbankknoten bezeichnet. Eine Umgebung mit partitionierten Datenbanken ist eine Datenbankinstallation, die die Verteilung von Daten für Datenbankpartitionen unterstützt.

Kapitel 7. Pufferpools

Ein *Pufferpool* ist ein Bereich im Hauptspeicher, der dem Datenbankmanager zum Zwischenspeichern von Tabellen- und Indexdaten beim Lesen von Daten vom Datenträger zugeordnet wurde. Jede DB2-Datenbank muss einen Pufferpool haben.

Jede neue Datenbank hat einen definierten Standardpufferpool mit dem Namen IBMDEFAULTBP. Zusätzliche Pufferpools können mithilfe der Anweisungen CREATE BUFFERPOOL, DROP BUFFERPOOL und ALTER BUFFERPOOL erstellt, gelöscht und geändert werden. Die Katalogsicht SYSCAT.BUFFERPOOLS greift auf die Informationen zu den Pufferpools zu, die in der Datenbank definiert sind.

In einer DB2 pureScale-Umgebung besitzt jedes Member seinen eigenen lokalen Pufferpool (LBP). Es ist jedoch ein zusätzlicher Gruppenpufferpool (GBP) vorhanden, der durch die Cluster-Caching-Funktion verwaltet wird. Der Gruppenpufferpool wird von allen Membern gemeinsam genutzt. Er wird als Cache für Seiten verwendet, die von einzelnen Membern in einer DB2 pureScale-Instanz verwendet werden, und soll die Leistung verbessern sowie die Konsistenz sicherstellen.

Verwendung von Pufferpools

Anmerkung: In den nachfolgenden Informationen werden Pufferpools in anderen Umgebungen als DB2 pureScale-Umgebungen behandelt. Pufferpools haben in DB2 pureScale-Umgebungen eine andere Funktionsweise.Weitere Informationen finden Sie im Abschnitt „Überwachung von Pufferpools in einer DB2 pureScale-Umgebung" in *Datenbanküberwachung - Handbuch und Referenz*.

Wenn auf eine Zeile von Daten in einer Tabelle zum ersten Mal zugegriffen wird, liest der Datenbankmanager die Seite, die diese Daten enthält, in einen Pufferpool ein. Seiten verbleiben im Pufferpool, bis die Datenbank gestoppt wird oder der von einer Seite belegte Speicherbereich im Pufferpool von einer anderen Seite benötigt wird.

Seiten im Pufferpool können entweder im Gebrauch sein oder nicht, und sie können genutzt (geändert) oder sauber (ungeändert) sein:

- Seiten, die im Gebrauch sind, werden momentan gelesen oder aktualisiert. Zur Erhaltung der Datenkonsistenz lässt der Datenbankmanager nur einen Agenten zu, der eine bestimmte Seite in einem Pufferpool zu einer bestimmten Zeit aktualisiert. Wenn eine Seite aktualisiert wird, wird auf sie ausschließlich durch einen einzigen Agenten zugegriffen. Wenn sie gelesen wird, kann sie von mehreren Agenten gleichzeitig gelesen werden.
- Genutzte Seiten enthalten Daten, die geändert, jedoch noch nicht auf die Platte geschrieben wurden.
- Wenn eine Seite auf die Platte geschrieben wurde, ist sie sauber und kann im Pufferpool bleiben.

Ein großer Teil der Optimierung einer Datenbank betrifft die Einstellung von Konfigurationsparametern, die das Einlesen von Daten in den Pufferpool und das Schreiben von Daten aus dem Pufferpool auf die Platte steuern. Wenn eine Seite nicht von einem zuletzt aktiven Agenten benötigt wurde, kann sie für neue Seitenanforderungen aus neuen Anwendungen verwendet werden. Die Leistung des Da-

tenbankmanagers kann durch zusätzliche Platten-E/A-Operationen beeinträchtigt werden.

Entwerfen von Pufferpools

Die Größen aller Pufferpools können eine große Auswirkung auf die Leistung Ihrer Datenbank haben.

Vor der Erstellung eines neuen Pufferpools müssen die folgenden Punkte geklärt werden:

- Welcher Pufferpoolname verwendet werden soll.
- Ob der Pufferpool sofort oder erst im Anschluss an die nächste Inaktivierung und erneute Aktivierung der Datenbank erstellt werden soll.
- Ob der Pufferpool für alle Datenbankpartitionen oder nur für eine Untergruppe der Datenbankpartitionen vorhanden sein soll.
- Welche Seitengröße für den Pufferpool verwendet werden soll. Siehe „Seitengrößen von Pufferpools" auf Seite 155.
- Ob der Pufferpool eine feste Größe haben soll oder ob der Datenbankmanager die Größe des Pufferpools als Reaktion auf Ihre Auslastung automatisch anpassen soll. Es wird empfohlen, den Pufferpool automatisch vom Datenbankmanager optimieren zu lassen, indem Sie den Parameter SIZE bei der Pufferpoolerstellung nicht angeben. Detaillierte Informationen finden Sie in der Beschreibung der Anweisung „CREATE BUFFERPOOL" sowie in „Hinweise zum Pufferpoolspeicher" auf Seite 155.
- Ob ein Teil des Pufferpools für die blockbasierte E/A reserviert werden soll. Detaillierte Informationen finden Sie in „Blockbasierte Pufferpools für besseren sequenziellen Vorablesezugriff".

Beziehung zwischen Tabellenbereichen und Pufferpools

Beim Entwerfen von Pufferpools müssen Sie mit der Beziehung zwischen Tabellenbereichen und Pufferpools vertraut sein. Jeder Tabellenbereich wird einem bestimmten Pufferpool zugeordnet. IBMDEFAULTBP ist der Standardpufferpool. Der Datenbankmanager ordnet darüber hinaus auch die folgenden Systempufferpools zu: IBMSYSTEMBP4K, IBMSYSTEMBP8K, IBMSYSTEMBP16K und IBMSYSTEMBP32K (bisher als „verdeckte Pufferpools" bezeichnet). Wenn ein anderer Pufferpool einem Tabellenbereich zugeordnet werden soll, muss der Pufferpool existieren und die gleiche Seitengröße aufweisen. Die Zuordnung wird bei der Erstellung des Tabellenbereichs (mit der Anweisung CREATE TABLESPACE) definiert. Zu einem späteren Zeitpunkt kann sie (mit der Anweisung ALTER TABLESPACE) geändert werden.

Durch die Verwendung mehrerer Pufferpools haben Sie die Möglichkeit, die Verwendung von Hauptspeicher durch die Datenbank zu konfigurieren, um die allgemeine Leistung zu verbessern. Zum Beispiel kann bei Tabellenbereichen mit einer oder mehreren umfangreichen Tabellen (d. h. größeren als der verfügbare Hauptspeicher), auf die die Benutzer wahlfrei zugreifen, die Größe des Pufferpools begrenzt werden, da ein Zwischenspeichern (Caching) der Datenseiten vielleicht nicht vorteilhaft ist. Dem Tabellenbereich für eine Online-Transaktionsanwendung kann ein größerer Pufferpool zugeordnet werden, sodass die Datenseiten, die von der Anwendung genutzt werden, länger im Cache zwischengespeichert und so schnellere Antwortzeiten erzielt werden können. Bei der Konfiguration neuer Pufferpools ist jedoch ein sorgfältiges Vorgehen geboten.

Seitengrößen von Pufferpools

Die Seitengröße für den Standardpufferpool wird bei der Verwendung des Befehls **CREATE DATABASE** definiert. Dieser Standardwert stellt die Standardseitengröße für alle zukünftigen Anweisungen CREATE BUFFERPOOL und CREATE TABLESPACE dar. Wenn Sie die Seitengröße beim Erstellen der Datenbank nicht angeben, wird die Standardseitengröße von 4 KB verwendet.

Anmerkung: Wenn Sie ermittelt haben, dass für Ihre Datenbank eine Seitengröße von 8 KB, 16 KB oder 32 KB erforderlich ist, müssen Sie mindestens einen Pufferpool mit der entsprechenden Seitengröße definiert und dem Tabellenbereich in Ihrer Datenbank zugeordnet haben.

Möglicherweise benötigen Sie jedoch einen Pufferpool, dessen Merkmale vom Systempufferpool abweichen. Sie können neue Pufferpools erstellen, die der Datenbankmanager verwendet. Sie müssen die Datenbank möglicherweise erneut starten, um die Änderungen an Tabellenbereichen und Pufferpools in Kraft zu setzen. Die Seitengrößen, die Sie für die Tabellenbereiche angeben, sollten die Seitengrößen bestimmen, die Sie für die Pufferpools auswählen. Die Auswahl der Seitengröße, die für einen Pufferpool angegeben wird, ist wichtig, da die Seitengröße nach der Erstellung des Pufferpools nicht mehr geändert werden kann.

Hinweise zum Pufferpoolspeicher

Speicherbedarf

Beim Entwerfen von Pufferpools müssen Sie auch den Speicherbedarf im Hinblick auf die auf Ihrem Computer installierte Speicherkapazität und im Verhältnis zu dem Speicher, der von anderen Anwendungen, die gleichzeitig mit dem Datenbankmanager auf demselben Computer ausgeführt werden, berücksichtigen. Datenauslagerungen (Swapping) durch das Betriebssystem treten auf, wenn nicht ausreichend Speicher zur Aufnahme aller Daten verfügbar ist, auf die zugegriffen wird. Dabei werden einige Daten in einen temporären Plattenspeicherbereich geschrieben, um im Hauptspeicher Platz für andere Daten zu schaffen. Wenn die Daten in dem temporären Plattenspeicherbereich benötigt werden, werden sie wieder zurück in den Hauptspeicher geladen.

Zugriffsschutz für Pufferpoolspeicher

In Version 9.5 werden Datenseiten im Pufferpoolspeicher mithilfe von Speicherschlüsseln geschützt. Diese Speicherschlüssel sind nur verfügbar, wenn sie durch die Registrierdatenbankvariable DB2_MEMORY_PROTECT explizit aktiviert werden. Dies gilt nur für das Betriebssystem AIX (5.3 TL06 5.4) bei Einsatz auf einem POWER6-Prozessor.

Pufferpoolspeicherschutz arbeitet auf Agentenbasis. Ein bestimmter Agent erhält nur dann Zugriff auf die Pufferpoolseiten, wenn dieser Agent den Zugriff benötigt. Der Speicherschutz funktioniert in der Weise, dass bestimmt wird, zu welchen Zeiten die Threads der DB2-Steuerkomponente Zugriff auf den Pufferpoolspeicher haben sollen und zu welchen sie keinen Zugriff haben sollen. Detaillierte Informationen finden Sie in „Zugriffsschutz für Pufferpoolspeicher (AIX auf POWER6)” auf Seite 159.

Address Windowing Extensions (AWE) und Extended Storage (ESTORE)

Anmerkung: Die AWE- und ESTORE-Komponenten, einschließlich der ESTORE-bezogenen Schlüsselwörter, Monitorelemente und Datenstrukturen, werden nicht mehr unterstützt. Wenn Sie über mehr Speicher verfügen

wollen, müssen Sie auf ein 64-Bit-Hardwarebetriebssystem mit den entsprechenden DB2-Produkten aufrüsten. Sie sollten außerdem Anwendungen und Scripts ändern, um Verweise auf diese nicht mehr unterstützte Funktionalität zu entfernen.

Pufferpools in einer DB2 pureScale-Umgebung

In einer DB2 pureScale-Umgebung stellt die Cluster-Caching-Funktion einen gemeinsamen Gruppenpufferpool (GBP, Group Buffer Pool) zur Verfügung, den alle Member gemeinsam nutzen. Jedes Member verwaltet darüber hinaus auch noch eine eigene Gruppe lokaler Pufferpools (LBPs, Local Buffer Pools).

Der Gruppenpufferpool (GBP) ist ein einzelner Pufferpool, der alle DB2-Seitengrößen unterstützt und den alle Member verwenden können. Member speichern Seiten in eigenen lokalen Pufferpools zwischen und verwenden den Gruppenpufferpool, um die Seitenkonsistenz zwischen Membern sicherzustellen. Auf jedem Member können lokale Pufferpools mit verschiedenen Seitengrößen (z. B. 4K, 8K, 16K oder 32K) vorhanden sein.

Der Gruppenpufferpool speichert zwei Typen von Informationen: Verzeichniseinträge und Datenelemente. Verzeichniseinträge speichern Metadateninformationen zu Pufferpoolseiten, Datenelemente speichern Seitendaten. Das Verhältnis zwischen Verzeichniseinträgen und Datenelementen wird von DB2 Database for Linux, UNIX and Windows automatisch angepasst. Der Gruppenpufferpoolspeicher wird durch den Konfigurationsparameter **cf_gbp_sz** definiert.

DB2-Pufferpoolservice

Da auf jedem Member lokale Pufferpools sowie ein Gruppenpufferpool, der von allen Membern verwendet wird, vorhanden sind, können mehrere Kopien derselben Seite in verschiedenen Pufferpools enthalten sein. Die globale Steuerung des gemeinsamen Zugriffs und der Kohärenz beim Zugriff auf die Seiten, um Änderungen an einer Seite auszuführen und Änderungen an andere Member weiterzugeben, werden vom DB2-Pufferpoolservice übernommen. Der Service verwaltet außerdem die Ein-/Ausgabe von Daten in Pufferpools, einschließlich des Schreibens von Seiten im Gruppenpufferpool auf die Platte.

GBP-Abhängigkeit

Eine Pufferpoolseite, auf die verschiedene Member in einer DB2 pureScale-Umgebung zugreifen müssen, ist GBP-abhängig. Für diese abhängigen Seiten koordiniert der Gruppenpufferpool die Kopien der Seite in verschiedenen Pufferpools. Eine Seite, auf die nur ein Member Zugriff hat, ist nicht GBP-abhängig und nur in einem lokalen Pufferpool des Members enthalten. In einer DB2 pureScale-Umgebung wird der Tabellenbereich für temporäre Tabellen nicht von verschiedenen Membern gemeinsam genutzt. D. h., die Pufferpoolseiten für den Tabellenbereich für temporäre Tabellen sind nicht GBP-abhängig.

P-Sperren (physische Sperren) steuern den Zugriff auf Pufferpoolseiten in einer DB2 pureScale-Umgebung für das Aktualisieren und Lesen einer Version einer Seite. Im Unterschied zu logischen Sperren (z. B. Zeilensperren oder Tabellensperren), die einer bestimmten Transaktion eigen sind, gehören P-Sperren, die den Zugriff auf Pufferpoolseiten steuern, Membern des Clusters. Die folgenden P-Sperren werden verwendet:

- Zum Aktualisieren einer Seite muss ein Member eine P-Sperre im exklusiven Modus (X-Modus) halten.

- Zum Lesen der neuesten Version einer Seite muss ein Member eine P-Sperre im Modus für gemeinsame Nutzung (S-Modus) halten.

Für das Lesen einer konsistenten Version, jedoch nicht notwendigerweise der neusten Version, der Seite, ist keine P-Sperre erforderlich. DB2 Database for Linux, UNIX and Windows entscheidet intern, welcher Lesetyp beim Zugriff auf eine Seite verwendet wird.

GBP-Kohärenz

Wenn eine Pufferpoolseite vom Gruppenpufferpool (GBP) abhängig ist, ist sie möglicherweise auf der Platte, im GBP, in einem lokalen Pufferpool (LBP) auf mehreren Membern oder an einer Kombination dieser Positionen vorhanden. Die folgenden Protokollregeln koordinieren die Kohärenz mehrerer Seiten:

- Wenn ein Member eine Seite in seinen lokalen Pufferpool abruft, registriert es sie beim Gruppenpufferpool.
- Bei dem Versuch, eine Seite in einen lokalen Pufferpool abzurufen, prüft ein Member zuerst den Gruppenpufferpool und liest den Inhalt der Seite nur dann von der Platte, wenn die Seite im Gruppenpufferpool nicht vorhanden ist. (Eine Version der Seite im Gruppenpufferpool ist nie älter als die Version der Seite auf der Platte.)
- Wenn ein Member eine Aktualisierungssperre für eine Seite (P-Sperre im X-Modus) hat, kann die Seite im lokalen Pufferpool des Members eine neuere Version als die Seite im Gruppenpufferpool haben.
- Bevor ein Member die Aktualisierungssperre freigibt (oder die P-Sperre herabstuft), wird der Gruppenpufferpool mit der neueren Version der Seite aktualisiert.
- Eine geänderte Seite wird in den Gruppenpufferpool geschrieben, wenn die Transaktion, die sie geändert hat, beendet wird (entweder durch eine COMMIT- oder eine ROLLBACK-Operation).
- Eine Seite kann außerdem vor dem Ende einer Transaktion durch eine Seitenvereinbarung in den Gruppenpufferpool geschrieben werden, wenn ein anderes Member eine P-Sperre anfordert, um die neueste Version zu lesen oder die Seite zu aktualisieren. Wenn ein anderes Member eine P-Sperre anfordert, hat der Sperrenkonflikt zur Folge, dass das Member, dem die Sperre gehört, eine geänderte Seite in den Gruppenpufferpool schreibt, sodass es die P-Sperre freigeben oder herabstufen kann. Anschließend kann die Sperre von dem anfordernden Member in Anspruch genommen werden.

Steuerung des Gruppenpufferpools

Die Gesamtspeichergröße, die vom Gruppenpufferpool (GBP) verwendet werden kann, wird durch den Datenbankkonfigurationsparameter **cf_gbp_sz** gesteuert. Der Gruppenpufferpool wird zugeordnet, wenn die Datenbank zum ersten Mal auf einem Member aktiviert wird, sofern der Gruppenpufferpool nicht bereits auf der CF vorhanden ist. Der Gruppenpufferpool wird freigegeben, wenn die CF gestoppt wird, wenn die Datenbank gelöscht oder im gesamten Cluster konsistent beendet wird oder wenn eine Restoreoperation für die Datenbank ausgeführt wird.

Das Castout schreibt Seiten aus dem GBP auf Platte und wird zwischen den Member koordiniert. Das Castout ist einer Seitenbereinigung in lokalen Pufferpools ähnlich und erfüllt zwei Funktionen:

- Auslagern genutzter (geänderter) Seiten auf die Platte und Sicherstellen, dass genügend saubere Verzeichniseinträge und Datenelemente vorhanden sind, die für neue Seitenregistrierungen und Seitenschreibvorgänge verwendet werden können.
- Bereitstellen eines bestimmten Recoveryfensters; dies geschieht, indem dafür gesorgt wird, dass sich im GBP nur Seiten bis zu einem bestimmten Alter befinden. Dadurch wird die Zahl der Protokollsätze reduziert, die im Falle einer Recovery wiederhergestellt werden müssen.

Falls erforderlich, können Sie das Castoutverhalten mit dem Datenbankkonfigurationsparameter **softmax** steuern. In einer DB2 pureScale-Umgebung bestimmt dieser Parameter, wie viele Seiten während der einzelnen Arbeitsphasen aus dem GDP auf die Platte geschrieben werden müssen.

Steuerung lokaler Pufferpools

Die Konfiguration der lokalen Pufferpools wird durch DDL-Anweisungen gesteuert. Das Member, durch das eine DDL-Anweisung ausgegeben wird, fungiert als Koordinator, der für die Verteilung der Anweisung an andere Member im Cluster zur lokalen Ausführung verantwortlich ist, und es koordiniert die Ausführung insgesamt. Das Ausführungsverhalten einer LBP-DDL-Anweisung in einer DB2 pureScale-Instanz weist deutliche Unterschiede gegenüber einer Instanz auf, die sich nicht in einer DB2 pureScale-Umgebung befindet. In einer DB2 pureScale-Umgebung müssen nicht alle Member (außer dem Koordinator), bei denen der LBP definiert ist, zu diesem Zeitpunkt verfügbar sein oder die Datenbank aktiviert haben, damit eine DDL-Anweisung erfolgreich ausgeführt wird. Bei Membern, die zu diesem Zeitpunkt nicht verfügbar sind (z. B. aufgrund einer planmäßigen Wartung) oder bei denen die Datenbank nicht aktiviert ist, wird die DDL-Anweisung nicht von ihnen selbst verarbeitet. Die DDL-Ausführung wird normal fortgesetzt. Wenn die Transaktion festgeschrieben wird, werden die Pufferpooländerungen vom Koordinator in den Pufferpooldateien auf der Platte für alle betroffenen Member aktualisiert, einschließlich derer, die die Anweisung nicht verarbeitet haben. Diese Member wenden die festgeschriebenen Pufferpooländerungen das nächste Mal an, wenn sie die Datenbank aktivieren. Wenn aber ein aktives Member die DDL-Anweisung nicht ausführt - weil nicht genug Speicher zur Verfügung steht oder wegen eines anderen Fehlers - wird die Anweisung per Rollback rückgängig gemacht und es wird ein Fehler an den Benutzer oder die Clientanwendung zurückgegeben.

Monitorelemente für Pufferpools

Sie können eine Reihe von Monitorelementen prüfen, die sich speziell auf den Gruppenpufferpool und lokale Pufferpools beziehen, um die Gesamtleistung von DB2 pureScale Feature zu überwachen. Diese Monitorelemente beziehen sich speziell auf den Datenbankkonfigurationsparameter **cf_gbp_sz**. Weitere Informationen zum Anzeigen von Speicherbelegungsniveaus für die Cluster-Caching-Funktion finden Sie im Abschnitt "MON_GET_CF (Tabellenfunktion) - Messdaten für Cluster-Caching-Funktion abrufen".

Darüber hinaus ist eine Reihe von Monitorelementen zur Verfolgung der Anzahlen physischer Seitenlesevorgänge, logischer Seitenlesevorgänge und gefundener gültiger Seiten für den Gruppenpufferpool und für logische Pufferpools verfügbar. Eine Liste der Monitorelemente für das DB2 pureScale Feature finden Sie unter "Neue und geänderte Monitorelemente".

Zugriffsschutz für Pufferpoolspeicher (AIX auf POWER6)

Der Datenbankmanager verwendet den Pufferpool, um Hinzufüge-, Änderungs- und Löschoperationen auf einen großen Teil der Datenbankdaten anzuwenden.

Speicherschlüssel sind ein neues Feature von IBM Power6-Prozessoren und des Betriebssystems AIX, das den Schutz von Speicherbereichen über Hardwareschlüssel auf der Kernel-Thread-Ebene ermöglicht. Der Schutz durch Speicherschlüssel reduziert Probleme aufgrund von Datenverlusten im Pufferpool und begrenzt Fehler, durch die die Datenbank gestoppt werden könnte. Unzulässige Versuche, durch Programmiertechniken auf den Pufferpool zuzugreifen, verursachen eine Fehlerbedingung, die der Datenbankmanager erkennen und behandeln kann.

Anmerkung: Pufferpoolspeicherschutz arbeitet auf Agentenbasis. Ein bestimmter Agent erhält nur dann Zugriff auf die Pufferpoolseiten, wenn dieser Agent den Zugriff benötigt.

Der Datenbankmanager schützt Pufferpools, indem er den Zugriff auf den Pufferpoolspeicher einschränkt. Wenn ein Agent Zugriff auf die Pufferpools benötigt, um seine Arbeit zu verrichten, wird ihm der Zugriff auf den Pufferpoolspeicher temporär erteilt. Wenn der Agent den Zugriff auf die Pufferpools nicht länger benötigt, wird der Zugriff entzogen. Durch dieses Verhalten wird sichergestellt, dass Agenten nur dann berechtigt werden, den Inhalt von Pufferpools zu ändern, wenn dies erforderlich ist. Dies verringert die Wahrscheinlichkeit von Datenverlusten in Pufferpools. Jeder unzulässige Zugriff auf den Pufferpoolspeicher führt zu einem Segmentierungsfehler. Zur Diagnose solcher Fehler stehen Tools zur Verfügung, wie zum Beispiel die Befehle **db2diag**, **db2fodc**, **db2pdcfg** und **db2support**.

Zur Aktivierung der Pufferpoolspeicherschutzfunktion zur Erhöhung der Ausfallsicherheit der Datenbanksteuerkomponente aktivieren Sie die Registrierdatenbankvariable **DB2_MEMORY_PROTECT**:

Registrierdatenbankvariable DB2_MEMORY_PROTECT

Diese Registrierdatenbankvariable aktiviert und inaktiviert die Schutzfunktion für den Pufferpoolspeicher. Wenn **DB2_MEMORY_PROTECT** aktiviert (auf YES gesetzt) ist und ein Thread der DB2-Steuerkomponente versucht, in unzulässiger Weise auf den Pufferpoolspeicher zuzugreifen, wird der Thread der Steuerkomponente abgefangen (Trap). Der Standardwert ist NO.

Hinweis:

Die Schutzfunktion für den Pufferpoolspeicher hängt von der Implementierung der AIX Storage Protect Keys (Speicherschutzschlüssel) ab und funktioniert mit dem fixierten gemeinsam genutzten Speicher möglicherweise nicht. Wenn **DB2_MEMORY_PROTECT** mit der Einstellung **DB2_PINNED_BP** oder **DB2_LARGE_PAGE_MEM** angegeben wird, werden AIX Storage Protect Keys möglicherweise nicht aktiviert. Weitere Informationen zu AIX Storage Protect Keys finden Sie unter http://publib.boulder.ibm.com/infocenter/systems/scope/aix/index.jsp?topic=/com.ibm.aix.genprogc/doc/genprogc/storage_protect_keys.htm.

Sie können keinen Speicherschutz verwenden, wenn **DB2_LGPAGE_BP** auf YES gesetzt ist. Selbst wenn **DB2_MEMORY_PROTECT** auf YES gesetzt ist, wird der DB2-Datenbankmanager den Pufferpoolspeicher nicht schützen und die Funktion inaktivieren.

Erstellen von Pufferpools

Mit der Anweisung CREATE BUFFERPOOL können Sie einen neuen Pufferpool definieren, der vom Datenbankmanager verwendet werden soll.

Vorbereitende Schritte

Es muss ausreichend Realspeicher auf dem Computer für die Gesamtgröße aller von Ihnen erstellten Pufferpools zur Verfügung stehen. Darüber hinaus benötigt auch das Betriebssystem einigen Speicher für seinen Betrieb.

Informationen zu diesem Vorgang

Bei partitionierten Datenbanken können auch definieren, dass der Pufferpool in der Datenbankpartition unterschiedlich, zum Beispiel mit unterschiedlichen Größen, erstellt wird. Mit der Standardklausel ALL DBPARTITIONNUMS wird der Pufferpool in allen Datenbankpartitionen der Datenbank erstellt.

Vorgehensweise

Gehen Sie wie folgt vor, um einen Pufferpool über die Befehlszeile zu erstellen:

1. Rufen Sie die Liste der Pufferpoolnamen, die in der Datenbank bereits vorhanden sind. Setzen Sie die folgende SQL-Anweisung ab:
   ```
   SELECT BPNAME FROM SYSCAT.BUFFERPOOLS
   ```
2. Wählen Sie einen Pufferpoolnamen aus, der sich momentan nicht in der Ergebnisliste befindet.
3. Legen Sie die Merkmale des zu erstellenden Pufferpools fest.
4. Vergewissern Sie sich, dass Sie die korrekte Berechtigungs-ID zum Ausführen der Anweisung CREATE BUFFERPOOL besitzen.
5. Führen Sie die Anweisung CREATE BUFFERPOOL aus. Eine einfache Anweisung CREATE BUFFERPOOL sieht folgendermaßen aus:
   ```
   CREATE BUFFERPOOL name_des_pufferpools
     PAGESIZE 4096
   ```

Ergebnisse

Wenn der Speicher ausreicht, ist der Pufferpool sofort aktiv. Standardmäßig werden neue Pufferpools mit dem Schlüsselwort IMMEDIATE erstellt. Auf den meisten Plattformen ist der Datenbankmanager in der Lage, mehr Speicher zuzuordnen. Das erwartete Rückgabeergebnis ist eine erfolgreiche Speicherzuordnung. Wenn der Datenbankmanager den zusätzlichen Speicher nicht zuordnen kann, gibt er eine Warnbedingung zurück, die besagt, dass der Pufferpool nicht gestartet werden konnte. Diese Warnung wird beim nachfolgenden Datenbankstart bereitgestellt. Für sofortige Anforderungen (IMMEDIATE) braucht die Datenbank nicht erneut gestartet zu werden. Wenn diese Anweisung festgeschrieben wird, wird der Pufferpool in die Systemkatalogtabellen eingetragen. Er wird jedoch erst aktiv, wenn die Datenbank das nächste Mal gestartet wird. Weitere Informationen zu dieser Anweisung, einschließlich weiterer Optionen, finden Sie in der Beschreibung der Anweisung „CREATE BUFFERPOOL".

Wenn Sie eine Anweisung CREATE BUFFERPOOL DEFERRED absetzen, wird der Pufferpool nicht sofort aktiviert, sondern erst beim nächsten Start der Datenbank erstellt. Bis zum erneuten Start der Datenbank verwenden alle neuen Tabellenbereiche einen vorhandenen Pufferpool, auch wenn dieser Tabellenbereich so erstellt

wird, dass er den verzögert (DEFERRED) erstellten Pufferpool explizit verwendet.

Beispiel

Im folgenden Beispiel gibt die optionale Klausel DATABASE PARTITION GROUP die Datenbankpartitionsgruppe (bzw. -gruppen) an, für die die Pufferpooldefinition gilt:

```
CREATE BUFFERPOOL name_des_pufferpools
  PAGESIZE 4096
  DATABASE PARTITION GROUP name_der_datenbankpartitionsgruppe
```

Wenn dieser Parameter angegeben wird, wird der Pufferpool nur in Datenbankpartitionen in diesen Datenbankpartitionsgruppen erstellt. Jede Datenbankpartitionsgruppe muss zu dem gegebenen Zeitpunkt in der Datenbank vorhanden sein. Wenn die Klausel DATABASE PARTITION GROUP nicht angegeben wird, wird dieser Pufferpool in allen Datenbankpartitionen (sowie in allen Datenbankpartitionen, die der Datenbank zu einem späteren Zeitpunkt hinzugefügt werden) erstellt.

Weitere Informationen finden Sie in der Beschreibung der Anweisung „CREATE BUFFERPOOL".

Ändern von Pufferpools

Es gibt eine Reihe von Gründen, aus denen es sinnvoll sein kann, einen Pufferpool zu ändern. Dazu gehört zum Beispiel die Aktivierung von Speicher mit automatischer Leistungsoptimierung. Zum Ändern von Pufferpools verwenden Sie die Anweisung ALTER BUFFERPOOL.

Vorbereitende Schritte

Die Berechtigungs-ID der Anweisung muss über die Berechtigung SYSCTRL oder SYSADM verfügen.

Informationen zu diesem Vorgang

Wenn Sie mit Pufferpools arbeiten, müssen Sie möglicherweise eine der folgenden Tasks ausführen:

- Aktivieren der automatischen Optimierung für einen Pufferpool, sodass der Datenbankmanager die Größe des Pufferpools als Reaktion auf die Anforderungen Ihrer Auslastung anpassen kann
- Ändern des Blockbereichs des Pufferpools zur Unterstützung der blockbasierten Ein-/Ausgabe
- Hinzufügen dieser Pufferpooldefinition zu einer neuen Datenbankpartitionsgruppe
- Ändern der Größe des Pufferpools in einigen oder allen Datenbankpartitionen

Zum Ändern eines Pufferpools über die Befehlszeile gehen Sie wie folgt vor:

1. Zum Abrufen der Liste der Pufferpoolnamen, die in der Datenbank bereits vorhanden sind, geben Sie die folgende Anweisung ein:

   ```
   SELECT BPNAME FROM SYSCAT.BUFFERPOOLS
   ```

2. Wählen Sie den Pufferpoolnamen in der Ergebnisliste aus.
3. Legen Sie fest, welche Änderungen vorgenommen werden müssen.
4. Vergewissern Sie sich, dass die korrekte Berechtigungs-ID zum Ausführen der Anweisung ALTER BUFFERPOOL vorhanden ist.

Anmerkung: Bei den Parametern `IMMEDIATE` und `DEFERRED` handelt es sich um Schlüsselparameter. Mit IMMEDIATE wird die Pufferpoolgröße geändert, ohne dass die nächste Datenbankaktivierung abgewartet werden muss, um die Änderung wirksam zu machen. Wenn nicht ausreichend gemeinsam verwendeter Datenbankspeicher vorhanden ist, um neuen Speicherbereich zuzuordnen, wird die Anweisung verzögert (DEFERRED) ausgeführt.

Bei Angabe von DEFERRED werden die Änderungen an dem Pufferpool erst angewendet, wenn die Datenbank erneut aktiviert wird. Reservierter Speicherbereich ist nicht erforderlich. Der Datenbankmanager ordnet den erforderlichen Speicher bei der Aktivierung automatisch aus dem System zu.

5. Mit der Anweisung ALTER BUFFERPOOL können Sie ein einzelnes Attribut des Pufferpoolobjekts ändern. Beispiel:

   ```
   ALTER BUFFERPOOL pufferpoolname SIZE anzahl_seiten
   ```

- Die Angabe *pufferpoolname* ist ein einteiliger Name, der einen Pufferpool angibt, der in den Systemkatalogen beschrieben ist.
- Die Angabe *anzahl_seiten* ist die neue Anzahl von Seiten, die diesem bestimmten Pufferpool zugeordnet werden soll. Sie können auch den Wert `-1` verwenden, um anzugeben, dass die Größe des Pufferpools dem Wert des Datenbankkonfigurationsparameters **buffpage** entsprechen soll.

Die Anweisung kann außerdem die Klausel DBPARTITIONNUM <datenbankpartitionsnummer> enthalten, um die Datenbankpartition anzugeben, in der die Größe des Pufferpools geändert werden soll. Wenn diese Klausel nicht angegeben wird, wird die Größe des Pufferpools in allen Datenbankpartitionen mit Ausnahme derer, die einen Ausnahmeeintrag in der Katalogsicht SYSCAT.BUFFERPOOLDBPARTITIONS haben, geändert. Detaillierte Informationen zur Verwendung dieser Klausel für Datenbankpartitionen finden Sie in der Beschreibung der Anweisung ALTER BUFFERPOOL.

Änderungen am Pufferpool durch diese Anweisung werden in die Systemkatalogtabellen eingetragen, wenn die Anweisung festgeschrieben (COMMIT) ist. Die Änderungen am Pufferpool werden jedoch erst wirksam, wenn die Datenbank das nächste Mal gestartet wird, sofern die Anweisung ALTER BUFFERPOOL nicht mit dem Standardschlüsselwort IMMEDIATE erfolgreich ausgeführt wurde.

Es muss ausreichend Realspeicher auf dem Computer für die Gesamtgröße aller von Ihnen erstellten Pufferpools zur Verfügung stehen. Darüber hinaus muss auch genügend Realspeicher für die übrigen Komponenten des Datenbankmanagers sowie für Ihre Anwendung vorhanden sein.

Löschen von Pufferpools

Stellen Sie beim Löschen von Pufferpools sicher, dass diesen Pufferpools keine Tabellenbereiche zugeordnet sind.

Der Pufferpool IBMDEFAULTBP kann nicht gelöscht werden.

Informationen zu diesem Vorgang

Der Plattenspeicher wird möglicherweise erst freigegeben, wenn die nächste Verbindung zur Datenbank hergestellt wird. Der Speicher eines gelöschten Pufferpools wird erst freigegeben, wenn die Datenbank gestoppt wird. Der Pufferpoolspeicher wird sofort zur Verwendung durch den Datenbankmanager freigegeben.

Vorgehensweise

Löschen Sie Pufferpools mit der Anweisung DROP BUFFERPOOL.

```
DROP BUFFERPOOL name_des_pufferpools
```

Kapitel 8. Tabellenbereiche

Ein *Tabellenbereich* ist eine Speicherstruktur, die Tabellen, Indizes, LOB-Daten und LONG-Daten enthält. Tabellenbereiche dienen zur Verwaltung von Daten in einer Tabelle in logischen Speichergruppierungen mit Bezug auf die Positionen, an denen Daten in einem System gespeichert werden. Tabellenbereiche werden in Datenbankpartitionsgruppen gespeichert.

Die Verwendung von Tabellenbereichen zur Speicherverwaltung bietet eine Reihe von Vorteilen:

Wiederherstellbarkeit
Wenn Objekte, die durch Backup gesichert bzw. durch Restore wiederhergestellt werden müssen, zusammen im selben Tabellenbereich angelegt werden, lassen sich Backup- und Restoreoperationen bequemer ausführen, da für alle Objekte in den Tabellenbereichen nur ein Backup- oder Restorebefehl ausgeführt werden muss. Wenn Sie partitionierte Tabellen und Indizes haben, die über Tabellenbereiche verteilt gespeichert werden, können Sie nur die Daten- und Indexpartitionen, die sich in einem bestimmten Tabellenbereich befinden, durch ein Backup sichern bzw. einen Restore wiederherstellen.

Mehr Tabellen
Es gibt Begrenzungen für die Anzahl von Tabellen, die in einem Tabellenbereich gespeichert werden können. Wenn Sie Bedarf an mehr Tabellen haben, als in einem Tabellenbereich gespeichert werden können, brauchen Sie nur zusätzliche Tabellenbereiche für diese Tabellen zu erstellen.

Speicherflexibilität
Bei DMS-Tabellenbereichen können Sie angeben, welche Speichereinheiten zum Speichern von Daten verwendet werden. Sie könnten zum Beispiel aktuelle, operative Daten in Tabellenbereichen auf schnelleren Einheiten speichern und Protokolldaten in Tabellenbereichen auf langsameren (und kostengünstigeren) Einheiten.

Möglichkeit zur Trennung von Daten in Pufferpools zur Verbesserung der Leistung und Speicherauslastung
Wenn Sie eine Gruppe von Objekten (z. B. Tabellen, Indizes) haben, die häufig abgefragt werden, können Sie dem Tabellenbereich, in dem sich diese befinden, durch eine Anweisung CREATE TABLESPACE oder ALTER TABLESPACE einen Pufferpool zuordnen. Sie können Tabellenbereiche für temporäre Daten einem eigenen Pufferpool zuordnen, um die Leistung von Aktivitäten wie Sortierungen oder Joins zu verbessern. In einigen Fällen kann es sinnvoll sein, kleinere Pufferpools für selten angeforderte Daten oder für Anwendungen, die einen sehr wahlfreien Zugriff auf eine sehr große Tabelle benötigen, zu definieren. In solchen Fällen brauchen Daten nicht länger als für eine einzige Abfrage erforderlich im Pufferpool behalten zu werden.

Tabellenbereiche bestehen aus mindestens einem *Container*. Ein Container kann ein Verzeichnisname, Einheitenname oder Dateiname sein. Ein einzelner Tabellenbereich kann mehrere Container haben. Mehrere Container (eines oder mehrerer Tabellenbereiche) können auf derselben physischen Speichereinheit erstellt werden (obwohl sich die beste Leistung erzielen lässt, wenn jeder Container, den Sie erstellen, eine andere Speichereinheit verwendet). Wenn Sie Tabellenbereiche mit dyna-

mischem Speicher verwenden, wird die Erstellung und Verwaltung von Containern automatisch durch den Datenbankmanager ausgeführt. Wenn Sie keine Tabellenbereiche mit dynamischem Speicher verwenden, müssen Sie Container selbst definieren und verwalten.

Abb. 7 zeigt ein Beispiel für die Beziehung zwischen Tabellen und Tabellenbereichen innerhalb einer Datenbank und den Containern, die dieser Datenbank zugeordnet sind.

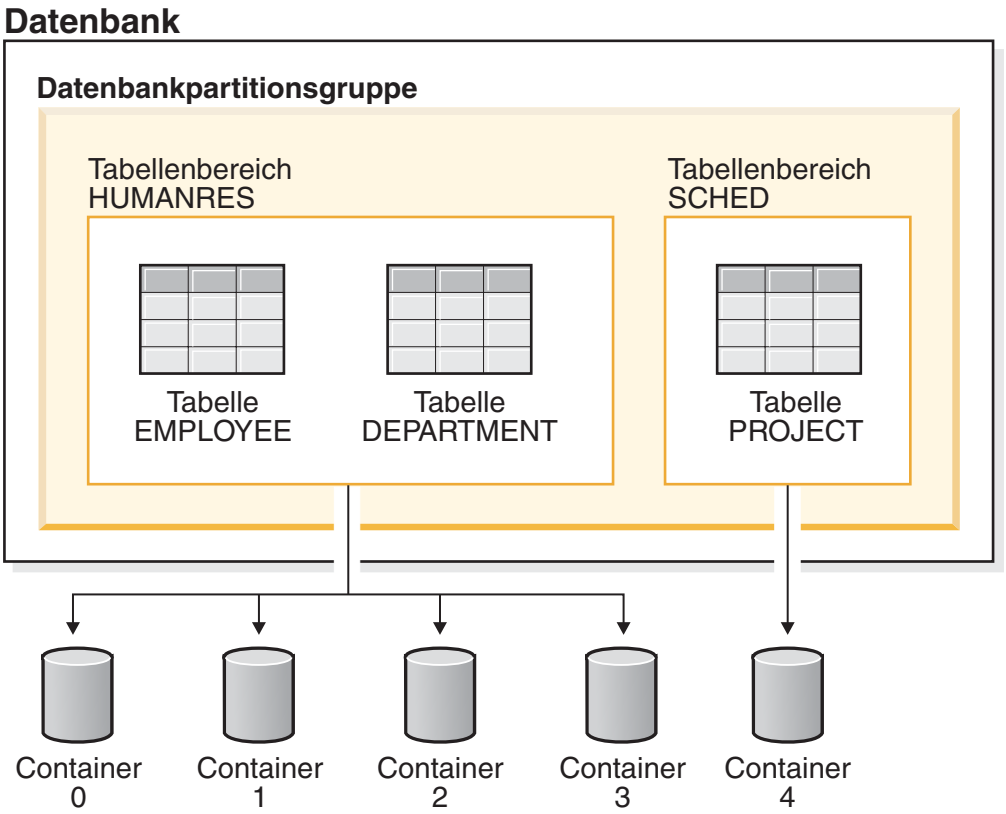


Abbildung 7. Tabellenbereiche und Tabellen in einer Datenbank

Die Tabellen EMPLOYEE und DEPARTMENT befinden sich im Tabellenbereich HUMANRES, der sich über die Container 0, 1, 2 und 3 erstreckt. Die Tabelle PROJECT befindet sich im Tabellenbereich SCHED in Container 4. In diesem Beispiel befindet sich jeder Container auf einer separaten Platte.

Der Datenbankmanager versucht, die Datenmenge möglichst gleichmäßig über die Container zu verteilen. Das heißt, es werden alle Container zum Speichern der Daten verwendet. Die Anzahl der Seiten, die der Datenbankmanager in einen Container schreibt, bevor er einen anderen Container verwendet, wird mit dem Parameter *EXTENTSIZE* definiert. Der Datenbankmanager beginnt beim Speichern der Tabellendaten nicht immer mit dem ersten Container.

Abb. 8 auf Seite 167 zeigt den Tabellenbereich HUMANRES mit einem Wert von zwei 4-KB-Seiten für EXTENTSIZE und vier Containern mit einer kleinen Anzahl zugeordneter Speicherbereiche. Die Tabellen DEPARTMENT und EMPLOYEE haben jeweils sieben Seiten und verteilen sich über alle vier Container.

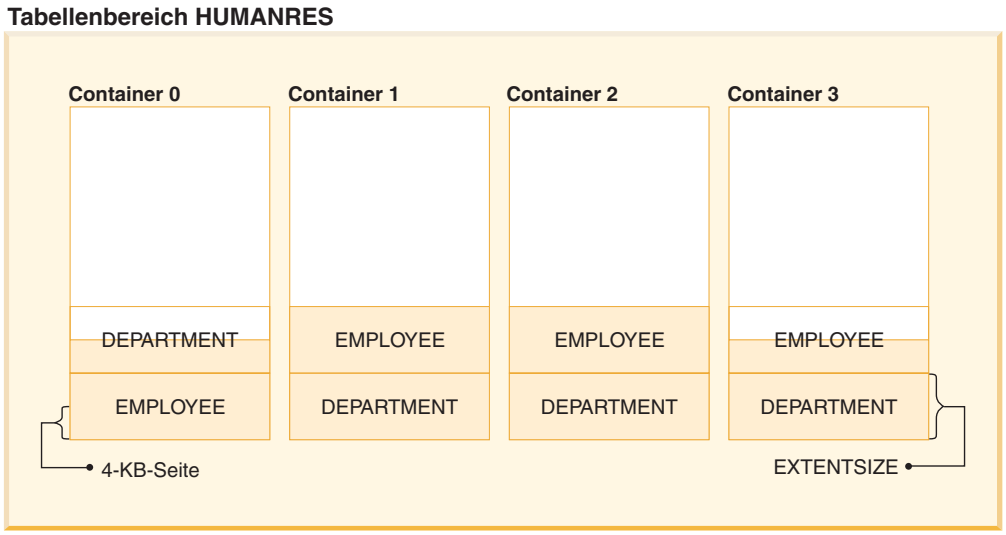


Abbildung 8. Container und EXTENTSIZE große Speicherbereiche in einem Tabellenbereich

Tabellenbereiche für Systemdaten, Benutzerdaten und temporäre Daten

Jede Datenbank muss über einen Mindestsatz an Tabellenbereichen verfügen, die zum Speichern von Systemdaten, Benutzerdaten und temporären Daten verwendet werden.

Eine Datenbank muss mindestens drei Tabellenbereiche enthalten:

- Einen *Katalogtabellenbereich*
- Mindestens einen *Tabellenbereich für Benutzertabellen*
- Mindestens einen *Tabellenbereich für temporäre Tabellen*

Katalogtabellenbereiche

Ein Katalogtabellenbereich enthält alle Systemkatalogtabellen für die Datenbank. Dieser Tabellenbereich heißt SYSCATSPACE und kann nicht gelöscht werden.

Tabellenbereiche für Benutzertabellen

Ein Tabellenbereich für Benutzertabellen enthält benutzerdefinierte Tabellen. Standardmäßig wird nur ein Tabellenbereich mit dem Namen USERSPACE1 erstellt.

Wenn Sie keinen Tabellenbereich für eine Tabelle bei der Erstellung angeben, wählt der Datenbankmanager einen Tabellenbereich aus. Weitere Informationen dazu finden Sie in der Dokumentation zur Klausel IN *tabellenbereichsname* der Anweisung CREATE TABLE.

Die Seitengröße eines Tabellenbereichs legt die maximale Zeilenlänge bzw. die Anzahl von Spalten fest, die Sie in einer Tabelle haben können. Die Dokumentation zur Anweisung CREATE TABLE enthält Informationen zur Beziehung zwischen der Seitengröße, der maximalen Zeilengröße und der Spaltenzahl. Vor Version 9.1 war die Standardseitengröße 4 KB. Ab Version 9.1 kann die Standardseitengröße einen der anderen unterstützten Werte haben. Die Standardseitengröße wird bei der

Erstellung einer neuen Datenbank deklariert. Auch mit deklarierter Standardseitengröße haben Sie weiterhin die Freiheit, einen Tabellenbereich mit einer Seitengröße für die Tabelle und einen anderen Tabellenbereich mit einer anderen Seitengröße für lange Daten oder LOB-Daten zu erstellen. Wenn die Anzahl der Spalten oder die Zeilengröße die Grenze für die Seitengröße eines Tabellenbereichs überschreitet, wird ein Fehler zurückgegeben (SQLSTATE 42997).

Tabellenbereiche für temporäre Tabellen

Ein Tabellenbereich für temporäre Tabellen enthält temporäre Tabellen. Tabellenbereiche für temporäre Tabellen können Tabellenbereiche für *temporäre Systemtabellen* oder *temporäre Benutzertabellen* sein.

Tabellenbereiche für temporäre Systemtabellen enthalten temporäre Daten, die der Datenbankmanager zur Ausführung von Operationen wie Sortierungen und Joins benötigt. Diese Typen von Operationen erfordern zusätzlichen Speicher zur Verarbeitung der Ergebnismengen. Eine Datenbank muss mindestens über einen Tabellenbereich für temporäre Systemtabellen verfügen. Standardmäßig wird ein Tabellenbereich für temporäre Systemtabellen mit dem Namen TEMPSPACE1 bei der Erstellung der Datenbank erstellt.

Bei der Verarbeitung von Abfragen benötigt der Datenbankmanager möglicherweise Zugriff auf einen Tabellenbereich für temporäre Systemtabellen mit einer Seitengröße, die ausreicht, um die zur Abfrage gehörenden Daten zu bearbeiten. Wenn eine Abfrage zum Beispiel Daten mit Zeilen zurückgibt, die 8 KB lang sind und keine Tabellenbereiche für temporäre Systemtabellen mit Seitengrößen von mindestens 8 KB vorhanden sind, könnte die Ausführung der Abfrage fehlschlagen. In diesem Fall müssen Sie möglicherweise einen Tabellenbereich für temporäre Systemtabellen mit einer größeren Seitengröße erstellen. Durch die Definition eines Tabellenbereichs für temporäre Tabellen mit einer Seitengröße, die der größten Seitengröße Ihrer Tabellenbereiche für Benutzertabellen entspricht, lassen sich solche Probleme besser vermeiden.

Tabellenbereiche für temporäre Benutzertabellen enthalten temporäre Daten aus Tabellen, die mit einer Anweisung DECLARE GLOBAL TEMPORARY TABLE oder CREATE GLOBAL TEMPORARY TABLE erstellt wurden. Sie werden nicht standardmäßig bei der Erstellung der Datenbank erstellt. Sie enthalten außerdem instanziierte Versionen von erstellten temporären Tabellen. Um die Definition deklarierter oder erstellter temporärer Tabellen zu ermöglichen, muss mindestens ein Tabellenbereich für temporäre Benutzertabellen mit den entsprechenden USE-Zugriffsrechten erstellt werden. USE-Zugriffsrechte werden mithilfe der Anweisung GRANT erteilt.

Wenn eine Datenbank mehr als einen temporären Tabellenbereich verwendet und ein neues temporäres Objekt benötigt wird, wählt das Optimierungsprogramm eine geeignete Seitengröße für dieses Objekt aus. Anschließend wird dieses Objekt dem temporären Tabellenbereich mit der entsprechenden Seitengröße zugeordnet. Wenn mehr als ein temporärer Tabellenbereich mit dieser Seitengröße vorhanden ist, werden die Tabellenbereiche reihum ausgewählt. Dabei wird mit einem Tabellenbereich dieser Seitengröße begonnen, anschließend der nächste für das nächste zuzuordnende Objekt ausgewählt und weiter, bis wieder der erste Tabellenbereich ausgewählt wird, nachdem alle geeigneten Tabellenbereiche verwendet wurden. In den meisten Fällen ist es jedoch nicht empfehlenswert, mehr als einen Tabellenbereich für temporäre Tabellen mit derselben Seitengröße zu haben.

Tabellenbereiche in einer Umgebung mit partitionierten Datenbanken

In einer Umgebung mit partitionierten Datenbanken wird jeder Tabellenbereich einer bestimmten Datenbankpartitionsgruppe zugeordnet. Dadurch können die Merkmale des Tabellenbereichs auf jede Datenbankpartition in der Datenbankpartitionsgruppe angewendet werden.

Wenn ein Tabellenbereich einer Datenbankpartitionsgruppe zugeordnet wird, muss die Datenbankpartitionsgruppe bereits vorhanden sein. Die Zuordnung zwischen dem Tabellenbereich und der Datenbankpartitionsgruppe wird bei der Erstellung des Tabellenbereichs mithilfe der Anweisung CREATE TABLESPACE definiert.

Sie können die Zuordnung zwischen einem Tabellenbereich und einer Datenbankpartitionsgruppe nicht ändern. Sie können lediglich die Tabellenbereichsspezifikation für einzelne Datenbankpartitionen innerhalb der Datenbankpartitionsgruppe mithilfe der Anweisung ALTER TABLESPACE ändern.

In einer Umgebung mit einer Einzelpartition wird jeder Tabellenbereich einer Standarddatenbankpartitionsgruppe wie folgt zugeordnet:

- Der Katalogtabellenbereich SYSCATSPACE wird der Gruppe IBMCATGROUP zugeordnet.
- Tabellenbereiche für Benutzertabellen werden der Gruppe IBMDEFAULTGROUP zugeordnet.
- Tabellenbereiche für temporäre Tabellen werden der Gruppe IBMTEMPGROUP zugeordnet.

In einer Umgebung mit partitionierten Datenbanken enthält die Partition IBMCATGROUP alle drei Standardtabellenbereiche, während die anderen Datenbankpartitionen nur die Tabellenbereiche TEMPSPACE1 und USERSPACE1 enthalten.

Hinweise zu Tabellenbereichen für DB2 pureScale Feature

Für eine DB2 pureScale-Umgebung sind bestimmte Typen von Tabellenbereichen erforderlich.

Unterstützung von Tabellenbereichstypen

In einer DB2 pureScale-Umgebung werden nur Tabellenbereiche mit dynamischem Speicher unterstützt. Vor der Migration von einer Umgebung, die keine DB2 pureScale-Umgebung ist, auf eine DB2 pureScale-Umgebung dürfen keine SMS-Tabellenbereiche (System Managed Space), regulären DMS-Tabellenbereiche (Database Managed Space) oder Hybridformen von Tabellenbereichen mit dynamischem Speicher vorhanden sein. Wenn der Befehl CREATE TABLESPACE für einen nicht dynamischen Speichertyp in einer DB2 pureScale-Umgebung abgesetzt wird, wird ein Fehler (SQL1419N) zurückgegeben.

Unterstützung von Tabellenbereichen für temporäre Tabellen

In einer DB2 pureScale-Konfiguration sind die Speicherpfade für Tabellenbereiche für temporäre Tabellen innerhalb des Clusterdateisystems definiert. Diese Tabellenbereiche für temporäre Tabellen werden von jedem Member lokal verwaltet. Der Containerpfad wird von allen Membern gemeinsam genutzt, und Tabellenbereichsdateien in diesem Verzeichnis werden jeweils eindeutig für jedes Member angelegt. Jedem Dateipfadnamen wird eine Memberkennung zugeordnet.

Transiente Statuswerte für Tabellenbereiche

In einer DB2 pureScale-Umgebung kann die Recovery nach dem Absturz eines Members parallel zu den normalen Laufzeitaktivitäten der anderen Members durchgeführt werden. Diese transienten Statussituationen müssen zum Blockieren anderen inkompatibler Operationen solange bestehen bleiben, bis die Recovery nach dem Absturz des Members abgeschlossen wurde. Es gibt zwei Typen von transienten Statussituationen für Tabellenbereiche. Der erste Typ kann jederzeit nach dem Ausfall des Members, jedoch vor dem Start der Recovery aufgehoben werden. Infolgedessen bleiben die Statuswerte SQLB_BACKUP_IN_PROGRESS, SQLB_DISABLE_PENDING oder SQLB_MOVE_IN_PROGRESS, wenn sie vom ausgefallenen Member festgelegt wurden, festgelegt, bis eines der folgenden Ereignisse eintritt:

1. Die Recovery nach Absturz eines Members für diese Datenbank wird eingeleitet. Oder:
2. Ein anderes Member muss diesen Tabellenbereich in seinen Speichercache laden.

Der andere Typ von transienten Statuswerten für Tabellenbereiche muss bis zum Ende der Recovery nach Absturz eines Members festgelegt bleiben, um die Operationen zu schützen, die zum Zeitpunkt des Fehlers ausgeführt wurden (z. B. eine Tabellenreorganisation und bestimmte Ladebefehle).

Wenn also der Statuswert SQLB_REORG_IN_PROGRESS oderSQLB_LOAD_IN-_PROGRESS vom ausgefallen Member festgelegt wird, bleibt er solange erhalten, bis die Recovery erfolgreich abgeschlossen ist.

Absetzen von DDL-Anforderungen für Tabellenbereiche

Wenn in einer Umgebung mit partitionierten Datenbanken der Host oder die DB2-Instanz einer Datenbankpartition inaktiv ist, schlägt eine DDL-Anweisung für Tabellenbereiche fehl und die Anweisung wird durch einen Rollback rückgängig gemacht. In dem Fall, dass der Host bzw. die DB2-Instanz aktiv ist, jedoch die Datenbankinfrastruktur dort nicht aktiviert ist, aktiviert die DDL-Anforderung die Datenbank implizit, damit die Anforderung verarbeitet wird. Diese allgemeinen Verhaltensweisen unterscheiden sich in einer DB2 pureScale-Umgebung wie folgt:

1. Für die Operationen **CREATE TABLESPACE**, **ALTER TABLESPACE** und **DROP TABLESPACE** ist es nicht erforderlich, dass alle Member erfolgreich abgeschlossen sind.
2. Wenn die Datenbankinfrastruktur auf einem Member nicht aktiviert ist, führt eine DDL-Anweisung für einen Tabellenbereich keine implizite Aktivierung aus.

Tabellenbereiche und Speicherverwaltung

Tabellenbereiche können je nach gewünschter Verwendungsweise des verfügbaren Speichers auf verschiedene Arten eingerichtet werden. Sie können das Betriebssystem die Zuordnung von Speicherressourcen überlassen oder Sie können den Datenbankmanager den Speicher für Ihre Daten nach Maßgabe von Parametern, die Sie angeben, zuordnen lassen. Oder können Sie Tabellenbereiche erstellen, die Speicher automatisch (dynamisch) zuordnen.

Es sind drei Typen von Tabellenbereichen verfügbar:

- SMS-Tabellenbereiche (SMS - System Managed Space), bei denen der Dateimanager des Betriebssystems den Speicher steuert, nachdem Sie die Position zum Speichern von Datenbankdateien definiert haben

- DMS-Tabellenbereiche (DMS - Database Managed Space), bei denen der Datenbankmanager die Nutzung des Speichers steuert, nachdem Sie die Speichercontainer zugeordnet haben
- Tabellenbereiche mit dynamischem Speicher, bei denen der Datenbankmanager die Erstellung von Containern nach Bedarf steuert

Jeder Typ von Tabellenbereich kann in beliebiger Kombination mit den anderen Typen in einer Datenbank verwendet werden.

Vom Betriebssystem verwalteter Speicherbereich (SMS)

In einem vom Betriebssystem verwalteten Tabellenbereich (SMS - System Managed Space) ordnet der Dateisystemmanager des Betriebssystems den Speicherbereich zu, in dem die Tabelle gespeichert wird, und verwaltet diesen Bereich. Im Gegensatz zu den vom Datenbankmanager verwalteten (DMS) Tabellenbereichen wird der Speicherplatz nicht im Voraus zugeordnet, wenn der Tabellenbereich erstellt wird. Er wird bei Bedarf zugeordnet.

Das SMS-Speichermodell besteht aus Dateien, die Datenbankobjekte darstellen. So hat jede Tabelle zum Beispiel mindestens eine physische Datei, die ihr zugeordnet ist. Wenn Sie den Tabellenbereich einrichten, legen Sie die Position der Dateien fest, indem Sie Container erstellen. Jedem Container in einem SMS-Tabellenbereich wird ein absoluter oder relativer Verzeichnisname zugeordnet. Jedes dieser Verzeichnisse kann sich auf einer anderen physischen Speichereinheit bzw. in einem anderen Dateisystem befinden. Der Datenbankmanager steuert die Benennung von Dateien, die für Objekte in den einzelnen Containern erstellt werden, während das Dateisystem für die Verwaltung der Dateien zuständig ist. Durch Steuern der Datenmenge, die in jede Datei geschrieben wird, verteilt der Datenbankmanager die Daten gleichmäßig auf die Container der Tabellenbereiche.

Wichtig: Benutzertabellenbereiche, die vom Betriebssystem verwaltete Speicherbereiche (System Managed Space, SMS) verwenden, sind veraltet und werden in einem zukünftigen Release möglicherweise entfernt. Katalogtabellenbereiche und temporäre Tabellenbereiche, die vom Betriebssystem verwaltete Speicherbereiche verwenden, sind nicht veraltet, es wird jedoch empfohlen, stattdessen vom Datenbankmanager verwaltete Tabellenbereiche (Database Managed Spaces, DMS) oder Tabellenbereiche mit dynamischem Speicher (Automatic Storage Spaces, AMS) zu verwenden.

Zuordnung von Speicherbereich

In einem SMS-Tabellenbereich wird der Speicher für Tabellen bei Bedarf zugeordnet. Die zugeordnete Speichergröße hängt von der Einstellung des Datenbankkonfigurationsparameters **multipage_alloc** ab. Wenn dieser Konfigurationsparameter auf den Wert YES (Standardwert) gesetzt ist, wird ein Speicherbereich in voller EXTENTSIZE-Größe (in der Regel mehrere Seiten) zugeordnet, wenn Speicherbereich benötigt wird. Ansonsten wird Speicherbereich jedes Mal in der Größe von einer Seite zugeordnet.

Die mehrseitige Dateizuordnung betrifft nur die Daten- und Indexteile einer Tabelle. Das heißt, dass die Dateien, die für lange Datentypen (LONG VARCHAR, LONG VAR GRAPHIC) und für große Objekte (LOBs) verwendet werden, nicht jedes Mal um einen EXTENTSIZE großen Speicherbereich erweitert werden.

Anmerkung: Die Zuordnung aus mehreren Seiten bestehender Dateien gilt nicht für Tabellenbereiche für temporäre Tabellen, die SMS-Speicherbereich verwenden.

Wenn der gesamte Speicherplatz in einem einzelnen Container in einem SMS-Tabellenbereich belegt wurde, wird der Tabellenbereich als voll betrachtet, selbst wenn noch weiterer Speicherplatz in anderen Containern verblieben ist. Im Unterschied zu DMS-Tabellenbereichen können einem SMS-Tabellenbereich nach der Erstellung keine Container hinzugefügt werden. Fügen Sie dem zugrunde liegenden Dateisystem mehr Speicherbereich hinzu, um zusätzlichen Speicherbereich für den SMS-Container bereitzustellen.

Planen von SMS-Tabellenbereichen

Wenn Sie die Verwendung von SMS-Tabellenbereichen erwägen, müssen Sie zwei Faktoren beachten:

- **Die Anzahl der benötigten Container für den Tabellenbereich.** Wenn Sie einen SMS-Tabellenbereich erstellen, müssen Sie die Anzahl von Containern angeben, die der Tabellenbereich verwenden soll. Es ist wichtig, alle gewünschten Container zu ermitteln, weil Sie nach dem Erstellen des Tabellenbereichs keine Container löschen oder hinzufügen können. Die einzige Ausnahme von dieser Regel tritt in einer Umgebung mit partitionierten Datenbanken auf. Wenn der Datenbankpartitionsgruppe eine neue Datenbankpartition für einen SMS-Tabellenbereich hinzugefügt wird, kann die Anweisung ALTER TABLESPACE verwendet werden, um der neuen Datenbankpartition Container hinzuzufügen.

  Die Maximalgröße des Tabellenbereichs kann anhand der folgenden Formel abgeschätzt werden:

  $n \times$ *maximale Dateisystemgröße*

  Dabei ist *n* die Anzahl von Containern und *maximale Dateisystemgröße* der Wert für die maximale Dateisystemgröße, die vom Betriebssystem unterstützt wird.

  Diese Formel geht davon aus, dass jedem Container ein unterschiedliches Dateisystem zugeordnet wird, dass jedes Dateisystem über den maximalen Speicherbereich verfügt und dass jedes Dateisystem dieselbe Größe hat. In der Praxis ist dies möglicherweise nicht der Fall, und die Maximalgröße des Tabellenbereichs kann sehr viel kleiner sein. Darüber hinaus bestehen auch Einschränkungen durch SQL bezüglich der Größe von Datenbankobjekten, die sich auf die maximale Größe eines Tabellenbereichs auswirken können.

  Achtung: Der Pfad, den Sie für den SMS-Tabellenbereich angeben, darf keine anderen Dateien oder Verzeichnisse enthalten.

- **Die Speicherbereichsgröße für den Tabellenbereich.** Die *Speicherbereichsgröße* (EXTENTSIZE) ist die Anzahl von Seiten, die der Datenbankmanager in einen Container schreibt, bevor er einen anderen Container verwendet. Die Speicherbereichsgröße kann nur beim Erstellen des Tabellenbereichs angegeben werden. Da spätere Änderungen daran nicht möglich sind, ist es wichtig, einen geeigneten Wert für die Speicherbereichsgröße anzugeben.

  Wenn Sie beim Erstellen eines Tabellenbereichs keinen Wert für die Speicherbereichsgröße angeben, erstellt der Datenbankmanager den Tabellenbereich mit dem Standardwert, der durch den Konfigurationsparameter **dft_extent_sz** der Datenbank definiert ist. Dieser Konfigurationsparameter wird anfangs auf der Grundlage der Informationen gesetzt, die beim Erstellen der Datenbank angegeben werden. Wenn der Wert für **dft_extent_sz** im Befehl **CREATE DATABASE** nicht angegeben wird, wird die Standardspeicherbereichsgröße auf den Wert 32 gesetzt.

Container und Speicherbereichsgröße

Um eine geeignete Anzahl von Containern und eine geeignete Speicherbereichsgröße für den Tabellenbereich festlegen zu können, benötigen Sie folgende Kenntnisse:

- **Den oberen Grenzwert, den Ihr Betriebssystem für die Größe eines logischen Dateisystems festlegt**. Zum Beispiel haben einige Betriebssysteme eine obere Begrenzung von 2 GB. Wenn Sie also ein Tabellenobjekt mit einer Größe von 64 GB erstellen möchten, benötigen Sie auf dieser Art System mindestens 32 Container. Wenn Sie einen Tabellenbereich erstellen, können Sie Container angeben, die sich auf verschiedenen Dateisystemen befinden, und dadurch die Menge der Daten erhöhen, die in der Datenbank gespeichert werden können.
- **Die Art und Weise, wie der Datenbankmanager die einem Tabellenbereich zugeordneten Datendateien und Container verwaltet**. Die erste Tabellendatendatei (konventionsgemäß `SQL00002.DAT`) wird in einem der Tabellenbereichscontainer erstellt. Der Datenbankmanager bestimmt diesen Container mithilfe eines Algorithmus, der die Gesamtzahl der Container zusammen mit der Tabellenkennung berücksichtigt. Die Datei kann bis zur Speicherbereichsgröße (EXTENTSIZE) anwachsen. Nach Erreichen dieser Größe schreibt der Datenbankmanager Daten in die Datei `SQL00002.DAT` im nächsten Container. Dieser Prozess wird fortgesetzt, bis alle Container Dateien des Namens `SQL00002.DAT` enthalten. Wenn dies der Fall ist, kehrt der Datenbankmanager zum Anfangscontainer zurück. Dieser Prozess, der auch als *Striping - einheitenübergreifendes Lesen und Schreiben von Daten* bezeichnet wird, wird über die Containerverzeichnisse fortgesetzt, bis ein Container voll ist (SQL0289N) oder vom Betriebssystem kein weiterer Speicherbereich mehr zugeordnet werden kann (Fehlernachricht: Datenträger voll). Das Striping betrifft die Blockzuordnungsdateien (`SQL`*`nnnnn`*`.BKM`), Indexobjekte und andere Objekte, die zum Speichern von Tabellendaten verwendet werden. Wenn Sie ein Platten-Striping zusammen mit dem vom Datenbankmanager bereitgestellten Striping implementieren, sollten die Speicherbereichsgröße (EXTENTSIZE) des Tabellenbereichs und die Stripegröße der Platte identisch sein.

  Anmerkung: Der SMS-Tabellenbereich wird als voll betrachtet, sobald irgendeiner seiner Container voll ist. Daher ist es wichtig, dass jedem Container dieselbe Menge an Speicherbereich zur Verfügung steht.

  SMS-Tabellenbereiche werden über die Option **`MANAGED BY SYSTEM`** im Befehl **`CREATE DATABASE`** oder in der Anweisung CREATE TABLESPACE definiert.

Von der Datenbank verwalteter Speicherbereich (DMS)

In einem DMS-Tabellenbereich (DMS = Database Managed Space) steuert der Datenbankmanager den Speicherbereich. Im Unterschied zu SMS-Tabellenbereichen wird der Speicherbereich im Dateisystem auf der Basis von Containerdefinitionen, die Sie bei der Erstellung des DMS-Tabellenbereichs angeben, vorab zugeordnet.

Das DMS-Speichermodell besteht aus einer begrenzten Anzahl von Dateien oder Einheiten, deren Speicherbereich vom Datenbankmanager verwaltet wird. Sie entscheiden, welche Dateien und Einheiten zu verwenden sind, wenn Sie Container erstellen, und Sie verwalten den Speicherbereich für diese Dateien und Einheiten.

Ein DMS-Tabellenbereich für benutzerdefinierte Tabellen und Daten kann als großer Tabellenbereich (*LARGE*, Standardtyp) oder als regulärer Tabellenbereich (*REGULAR*) definiert werden, in dem beliebige Tabellendaten oder Indexdaten gespeichert werden. Die Maximalgröße eines regulären Tabellenbereichs beträgt 512 GB bei 32-KB-Seiten. Die Maximalgröße eines großen Tabellenbereichs beträgt 64 TB. In „SQL- und XML-Begrenzungen" im Handbuch *SQL Reference* finden Sie Informationen zur maximalen Größe regulärer Tabellenbereiche für andere Seitengrößen.

Es stehen zwei Optionen für Container zur Verfügung, wenn Sie mit DMS-Tabellenbereichen arbeiten: Dateien und Roheinheiten (Raw Devices). Beim Arbeiten mit Dateicontainern ordnet der Datenbankmanager den gesamten Container bei der Er-

stellung des Tabellenbereichs zu. Ein Ergebnis dieser ersten Zuordnung des gesamten Tabellenbereichs besteht darin, dass die physische Zuordnung zwar nicht garantiert, jedoch normalerweise zusammenhängend erfolgt, obwohl das Dateisystem die Zuordnung vornimmt. Beim Arbeiten mit Containern für Roheinheiten übernimmt der Datenbankmanager die Steuerung der gesamten Einheit und stellt immer sicher, dass die Seiten in einem EXTENTSIZE großen Speicherbereich (*Extent*) zusammenhängen. (Ein EXTENTSIZE großer *Speicherbereich* ist als die Anzahl von Seiten definiert, die der Datenbankmanager in einen Container schreibt, bevor er einen anderen Container verwendet.)

Planen von DMS-Tabellenbereichen

Beachten Sie beim Entwerfen Ihrer DMS-Tabellenbereiche und Container Folgendes:

- Der Datenbankmanager arbeitet mit einheitenübergreifendem Lesen und Schreiben von Daten (Striping), um eine gleichmäßige Verteilung von Daten auf alle Container sicherzustellen. Dadurch werden die Daten gleichmäßig in alle Container im Tabellenbereich geschrieben, wobei die EXTENTSIZE großen Speicherbereiche für Tabellen reihum in allen Containern angelegt werden. Das DB2-Striping wird empfohlen, wenn Daten in mehrere Container geschrieben werden. Wenn Sie ein Platten-Striping zusammen mit dem DB2-Striping implementieren, sollten der Wert für EXTENTSIZE des Tabellenbereichs und die Stripegröße der Platte identisch sein.
- Im Unterschied zu SMS-Tabellenbereichen müssen die Container, die einen DMS-Tabellenbereich bilden, nicht die gleiche Größe haben. Dies wird im Normalfall jedoch nicht empfohlen, da es zu ungleichmäßiger Verteilung (Striping) auf die Container sowie zu nicht optimaler Leistung führt. Wenn ein Container voll ist, wird in DMS-Tabellenbereichen jeder verfügbare freie Speicherbereich anderer Container genutzt.
- Da der Speicherbereich vorab zugeordnet wird, muss er zur Verfügung stehen, bevor der Tabellenbereich erstellt werden kann. Bei Verwendung von Einheitencontainern muss die Einheit ebenfalls mit genügend Speicherbereich für die Definition des Containers verfügbar sein. Auf jeder Einheit kann nur ein Container definiert werden. Um eine Verschwendung von Speicherbereich zu vermeiden, sollten die Größe der Einheit und die Größe des Containers äquivalent sein. Wenn zum Beispiel die Einheit eine Speicherkapazität äquivalent zu 5000 Seiten hat und der Einheitencontainer mit einer Größe von 3000 Seiten definiert wird, sind 2000 Seiten auf der Einheit nicht verwendbar.
- Standardmäßig wird ein EXTENTSIZE großer Speicherbereich (Extent) in jedem Container für zusätzlichen Speicherbedarf reserviert. Nur ganze, durch EXTENTSIZE definierte Speicherbereiche werden verwendet. Für eine optimale Speicherverwaltung können Sie daher beim Zuordnen eines Containers eine geeignete Größe anhand der folgenden Formel bestimmen:

  $$extentsize * (n + 1)$$

  Dabei ist *extentsize* die Größe jedes EXTENTSIZE großen Speicherbereichs im Tabellenbereich, und *n* ist die Anzahl dieser Speicherbereiche, die Sie in dem Container speichern wollen.
- Die Mindestgröße eines DMS-Tabellenbereichs beträgt fünf EXTENTSIZE große Speicherbereiche.
  - Drei Speicherbereiche im Tabellenbereich sind für den Systemaufwand reserviert.

– Mindestens zwei Speicherbereiche sind erforderlich, um Tabellendaten des Benutzers zu speichern. (Diese zwei Speicherbereiche sind für die regulären Daten einer Tabelle vorgesehen, und nicht für Index-, Langfeld- oder LOB-Daten, die eigene Speicherbereiche benötigen.)

Jeder Versuch, einen Tabellenbereich zu erstellen, der kleiner als fünf EXTENTSIZE-Bereiche ist, führt zu einem Fehler (SQL1422N).

- Einheitencontainer müssen logische Datenträger mit einer „zeichenspezifischen Schnittstelle" (d. h. keine physischen Datenträger) verwenden.
- Für DMS-Tabellenbereiche können auch Dateien anstelle von Einheiten (devices) verwendet werden. Mit dem Standardtabellenbereichsattribut NO FILE SYSTEM CACHING in Version 9.5 können Dateien eine ähnliche Leistung wie Einheiten erreichen. Der Vorteil ist, dass keine Einheiten eingerichtet werden müssen. Weitere Informationen hierzu finden Sie in „Tabellenbereiche ohne Dateisystemcaching" auf Seite 209.
- Wenn die Auslastung LOB- oder LONG VARCHAR-Daten umfasst, kann die Verwendung des Dateisystemcache Leistungsvorteile erbringen.

  Anmerkung: LOB- und LONG VARCHAR-Daten werden nicht vom Pufferpool des Datenbankmanagers zwischengespeichert (gepuffert).
- Einige Betriebssysteme erlauben den Betrieb physischer Einheiten, die größer als 2 GB sind. Sie sollten in Betracht ziehen, die physische Einheit in logische Einheiten zu unterteilen, damit kein Container größer als die vom Betriebssystem zugelassene Größe ist.

Beim Arbeiten mit DMS-Tabellenbereichen sollten Sie in Betracht ziehen, jeden Container einer anderen Platte zuzuordnen. Dadurch wird die Tabellenbereichskapazität vergrößert und die Möglichkeit geschaffen, parallele E/A-Operationen zu nutzen.

Die Anweisung CREATE TABLESPACE erstellt einen neuen Tabellenbereich in einer Datenbank, ordnet diesem Tabellenbereich Container zu und trägt die Definition und die Attribute des Tabellenbereichs in den Katalog ein. Wenn Sie einen Tabellenbereich erstellen, wird EXTENTSIZE als Anzahl zusammenhängender Seiten definiert. Nur jeweils eine Tabelle oder ein Objekt, zum Beispiel ein Index, kann die Seiten in einem einzelnen Speicherbereich verwenden. Allen Objekten, die in dem Tabellenbereich erstellt werden, werden Speicherbereiche in einer logischen Adressenzuordnung des Tabellenbereichs zugeordnet. Die Zuordnung von Speicherbereichen wird über Speicherzuordnungsseiten (SMP = Space Map Pages) verwaltet.

Der erste Speicherbereich in der logischen Adressenzuordnung für den Tabellenbereich besteht aus den Kopfdaten für den Tabellenbereich, die interne Steuerdaten enthalten. Der zweite Speicherbereich ist der erste Speicherbereich der *Speicherzuordnungsseiten* (SMP) für den Tabellenbereich. SMP-Speicherbereiche sind in regelmäßigen Abständen über den gesamten Tabellenbereich verteilt. Jeder SMP-Speicherbereich besteht aus einer Bitzuordnung der Speicherbereiche vom aktuellen SMP-Speicherbereich bis zum nächsten SMP-Speicherbereich. Die Bitzuordnung dient zur Verfolgung, welche der dazwischenliegenden Speicherbereiche in Gebrauch sind.

Der auf die SMP folgende Speicherbereich ist die Objekttabelle für den Tabellenbereich. Die Objekttabelle ist eine interne Tabelle, die aufzeichnet, welche Benutzerobjekte im Tabellenbereich vorhanden sind und wo sich deren erster EMP-Speicherbereich (EMP = Extent Map Page, Speicherbereichsmaskenseite) befindet. Jedes Objekt verfügt über eigene EMPs, die eine Maske zu allen Seiten des Objekts dar-

stellen, das in der logischen Adressenzuordnung für den Speicherbereich gespeichert ist. Abb. 9 zeigt, wie Speicherbereiche in einer logischen Tabellenbereichsadressenzuordnung zugeordnet werden.

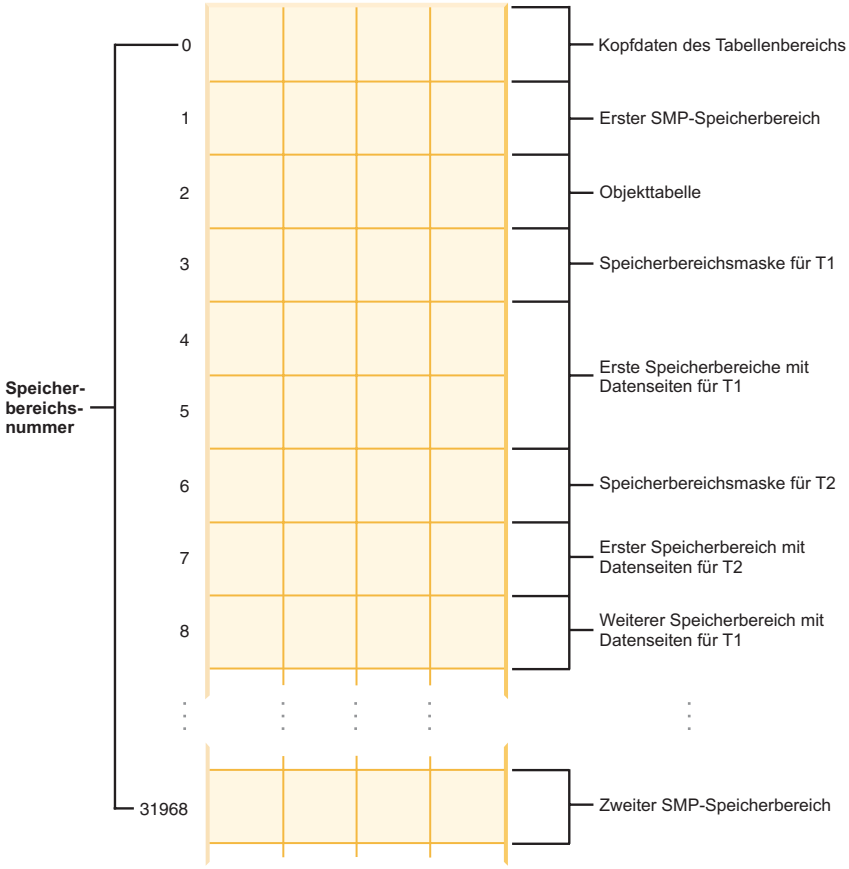


Abbildung 9. Logische Tabellenbereichsadresszuordnung

Tabellenbereichszuordnungen für DMS-Tabellenbereiche:

Eine Tabellenbereichszuordnung ist im Datenbankmanager eine interne Darstellung eines DMS-Tabellenbereichs, die die Konvertierung logischer Seitenpositionen in physische Seitenpositionen in einem Tabellenbereich beschreibt. Dieser Abschnitt erläutert die Nützlichkeit einer Tabellenbereichszuordnung und die Herkunft der Informationen in einer Tabellenbereichszuordnung.

In einer partitionierten Datenbank werden Seiten in einem DMS-Tabellenbereich logisch von 0 bis (*N*-1) durchnummeriert, wobei *N* die Anzahl verwendbarer Seiten im Tabellenbereich darstellt.

Die Seiten in einem Tabellenbereich werden zu Speicherbereichen gruppiert, deren Größe durch den Parameter EXTENTSIZE definiert ist. Aus Sicht der Tabellenbereichsverwaltung erfolgt jede Zuordnung von Objekten auf der Grundlage der durch EXTENTSIZE definierten Speicherbereiche. Auf diese Weise ist es möglich, dass eine Tabelle vielleicht nur die Hälfte der Seiten in einem EXTENTSIZE-Speicherbereich verwendet, jedoch wird der gesamte Speicherbereich als in Gebrauch und dem jeweiligen Objekt zugeordnet betrachtet. Standardmäßig wird ein EXTENTSIZE großer Speicherbereich zur Aufnahme der Containerkennung verwendet, wobei dieser Speicherbereich nicht zum Speichern von Daten verwendet

werden kann. Wenn jedoch die Registrierdatenbankvariable **DB2_USE_PAGE_CONTAINER_TAG** aktiviert ist, wird nur eine Seite für die Containerkennung verwendet.

In Abb. 10 sehen Sie die logische Adressenzuordnung für einen DMS-Tabellenbereich.

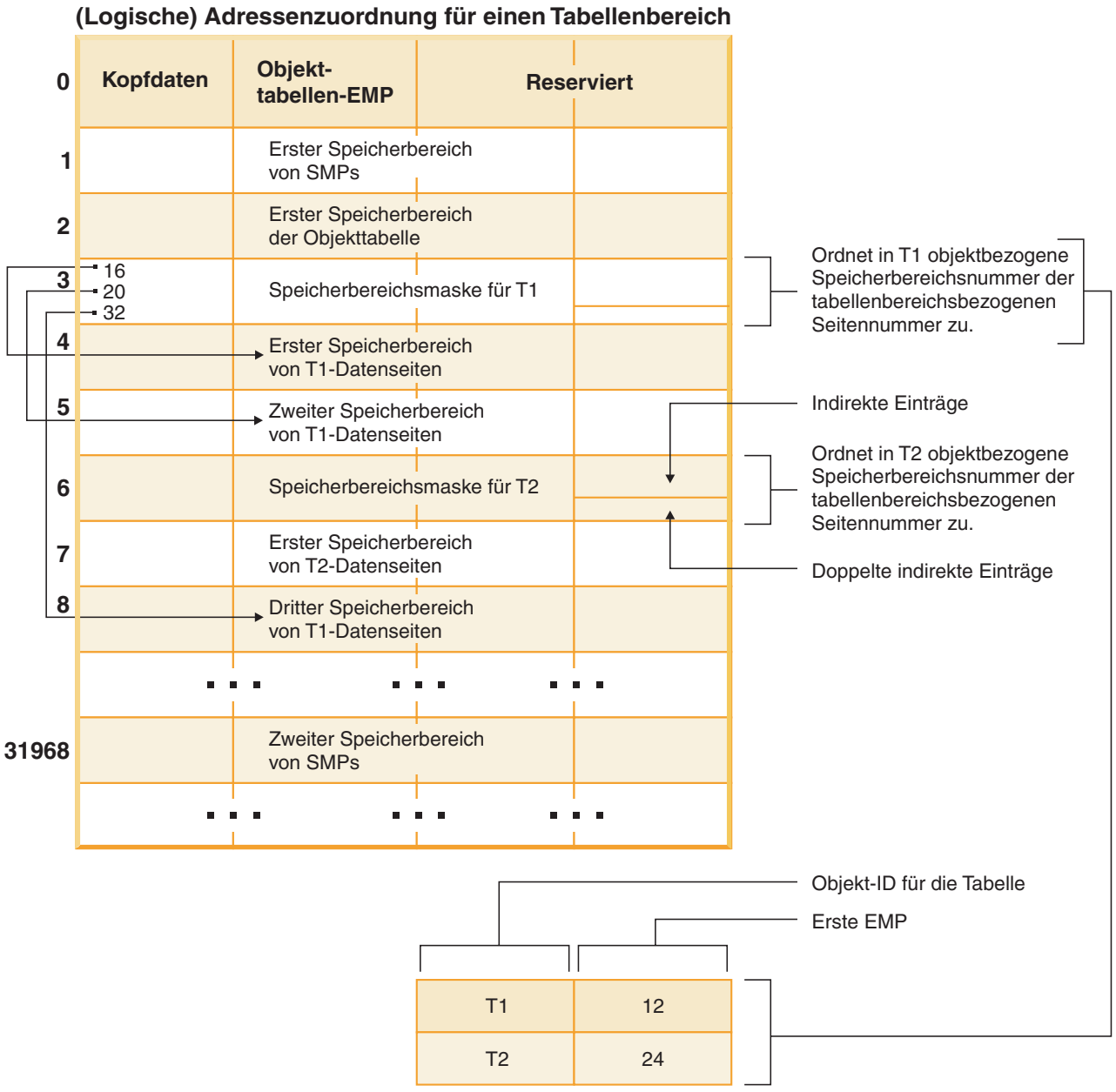


Abbildung 10. DMS-Tabellenbereiche

In der Adressenzuordnung des Tabellenbereichs gibt es zwei Typen von Zuordnungsseiten: EMP-Seiten (Extent Map Pages) und SMP-Seiten (Space Map Pages).

Die Objekttabelle ist eine interne relationale Tabelle, die eine Objektkennung der Position des ersten EMP-Speicherbereichs in der Tabelle zuordnet. Dieser EMP-Speicherbereich bildet, direkt oder indirekt, eine Zuordnung aller Speicherbereiche im betreffenden Objekt. Jede EMP enthält eine Reihe von Einträgen. Jeder Eintrag

ordnet eine zum Objekt relative Speicherbereichsnummer einer zum Tabellenbereich relativen Seitennummer zu, an der sich der Objektspeicherbereich befindet. Direkte EMP-Einträge ordnen Adressen, die zum Objekt relativ sind, direkt Adressen zu, die zum Tabellenbereich relativ sind. Die letzte EMP-Seite im ersten EMP-Speicherbereich enthält indirekte Einträge. Indirekte EMP-Einträge ordnen EMP-Seiten zu, die anschließend Zuordnungen zu Objektseiten herstellen. Die letzten 16 Einträge in der letzten EMP-Seite im ersten EMP-Speicherbereich enthalten doppelt indirekte Einträge.

Die Speicherbereiche der Adressenzuordnung für den logischen Tabellenbereich werden reihum einheitenübergreifend in den Containern gespeichert, die dem Tabellenbereich zugeordnet sind.

Da der Speicher in Containern jeweils in Form eines EXTENTSIZE großen Bereichs zugeordnet wird, werden Seiten, die nicht eine volle EXTENTSIZE-Größe bilden, nicht verwendet. Wenn Sie zum Beispiel einen 205 Seiten großen Container mit einem EXTENTSIZE-Wert 10 haben, wird ein EXTENTSIZE-Speicherbereich für die Kennung verwendet und 19 EXTENTSIZE-Speicherbereiche sind für Daten verfügbar. Die fünf übrigen Seiten werden verschenkt.

Wenn ein DMS-Tabellenbereich einen einzelnen Container enthält, ist die Umwandlung der logischen Seitennummer in die physische Position auf dem Datenträger ein einfacher Prozess, bei dem die Seiten 0, 1, 2 in der gleichen Reihenfolge auf dem Datenträger angeordnet werden.

Ebenfalls recht einfach ist der Prozess, wenn mehr als ein Container vorhanden ist und jeder der Container gleich groß ist. Der erste EXTENTSIZE-Speicherbereich im Tabellenbereich, der die Seiten 0 bis (EXTENTSIZE - 1) enthält, wird im ersten Container angelegt, der zweite EXTENTSIZE-Speicherbereich wird im zweiten Container angelegt usw. Nach dem letzten Container wird der Prozess wiederholt, wobei wieder mit dem ersten Container begonnen wird. Durch diesen zyklischen Prozess werden die Daten gleichmäßig verteilt.

Für Tabellenbereiche, die Container unterschiedlicher Größen enthalten, kann kein einfaches Reihumverfahren angewandt werden, da in diesem Fall der zusätzliche Speicherplatz in den größeren Containern nicht genutzt wird. An dieser Stelle kommt die Tabellenbereichszuordnung (engl. table space map) ins Spiel: Sie gibt an, wie die EXTENTSIZE-Speicherbereiche innerhalb des Tabellenbereichs positioniert sind, und stellt dadurch sicher, dass alle Speicherbereiche in den physischen Containern zur Verwendung verfügbar sind.

Anmerkung: In den folgenden Beispielen wird bei den Containergrößen die Größe der Containerkennung (container tag) nicht berücksichtigt. Die Containergrößen sind sehr klein und dienen lediglich zu Veranschaulichungszwecken. Sie stellen keine empfohlenen Containergrößen dar. Die Beispiele zeigen Container unterschiedlicher Größen innerhalb eines Tabellenbereichs, jedoch wird empfohlen, Container gleicher Größe zu verwenden.

Beispiel 1:

In einem Tabellenbereich sind drei Container vorhanden, jeder Container enthält 80 verwendbare Seiten und der EXTENTSIZE-Wert für den Tabellenbereich beträgt 20. Jeder Container enthält daher vier EXTENTSIZE große Speicherbereiche (80 / 20) mit insgesamt zwölf Speicherbereichen. Diese Speicherbereiche (Extents) befinden sich auf dem Datenträger wie in Abb. 11 auf Seite 179 gezeigt.

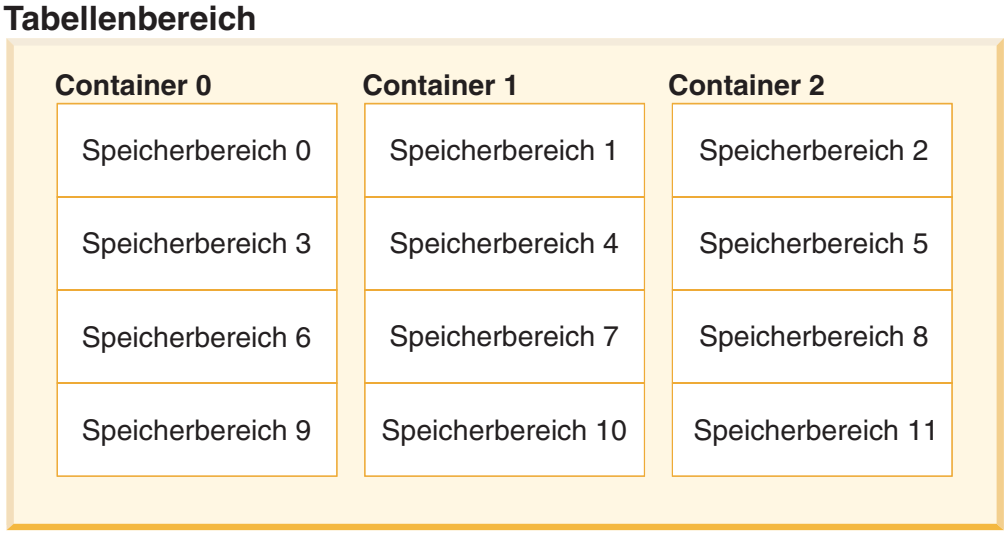


Abbildung 11. Tabellenbereich mit drei Containern und zwölf Speicherbereichen

Wenn Sie sich eine Tabellenbereichszuordnung ansehen wollen, erstellen Sie mithilfe des Snapshot Monitor eine Momentaufnahme des Tabellenbereichs. In Beispiel 1, in dem die drei Container die gleiche Größe besitzen, sieht die Tabellenbereichszuordnung wie folgt aus:

```
Range   Stripe Stripe Max      Max   Start   End    Adj.  Containers
Number  Set    Offset Extent   Page  Stripe  Stripe
   [0]    [0]       0     11    239       0       3   0   3 (0, 1, 2)
```

Ein *Bereich* (engl. *range*) ist der Teil der Zuordnung, in dem ein zusammenhängender Bereich von Stripes jeweils die gleiche Gruppe von Containern enthalten. In Beispiel 1 enthalten alle Stripes (0 - 3) die gleiche Gruppe von drei Containern (0, 1 und 2), sodass dies als ein einzelner Bereich betrachtet wird.

Die Spaltenüberschriften in der Tabellenbereichszuordnung heißen 'Range Number' (Bereichsnummer), 'Stripe Set', 'Stripe Offset', 'Maximum extent number addressed by the range' (höchste Speicherbereichsnummer, die durch den Bereich adressiert wird), 'Maximum page number addressed by the range' (höchste Seitennummer, die durch den Bereich adressiert wird), 'Start Stripe' (Anfangsstripe), 'End Stripe' (Endstripe), 'Range adjustment' (Bereichsanpassung) und 'Container list' (Containerliste). Diese Namen werden in Beispiel 2 detaillierter beschrieben.

Dieser Tabellenbereich kann auch wie in Abb. 12 auf Seite 180 dargestellt werden, wobei jede vertikale Linie einem Container entspricht und jede horizontale Linie als *Stripe* (einheitenübergreifend gespeicherter Datenblock) bezeichnet wird. Jede Zellennummer entspricht einem EXTENTSIZE großen Speicherbereich (Extent).

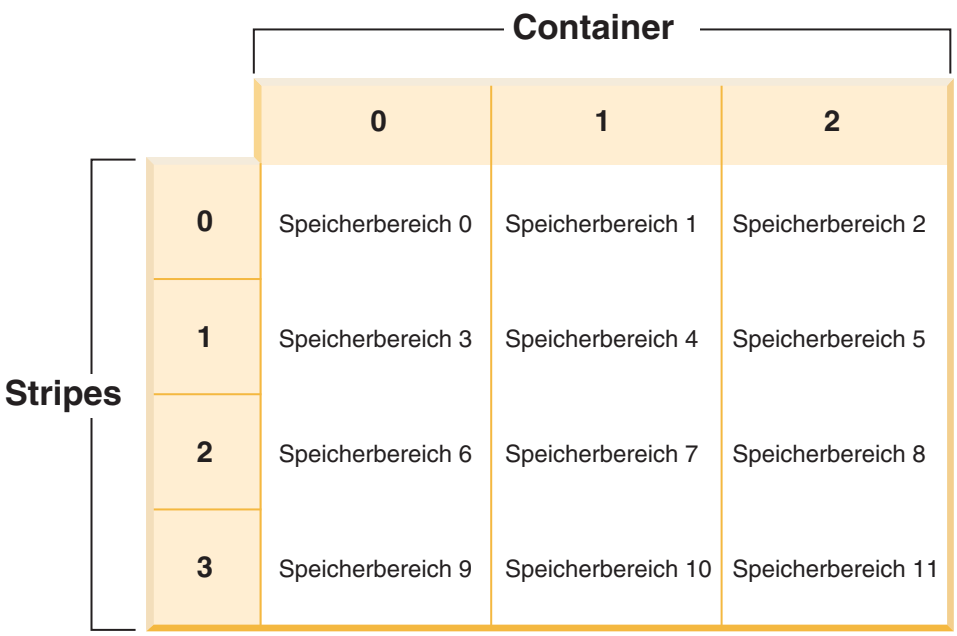


Abbildung 12. Tabellenbereich mit drei Containern und zwölf Speicherbereichen, Stripes hervorgehoben

Beispiel 2:

Im Tabellenbereich sind zwei Container vorhanden: der erste ist 100 Seiten groß, der zweite 50 Seiten, und EXTENTSIZE definiert eine Größe von 25 Seiten. Dies bedeutet, dass der erste Container vier EXTENTSIZE große Speicherbereiche und der zweite Container zwei solche Speicherbereiche besitzt. Der Tabellenbereich lässt sich wie in Abb. 13 darstellen.

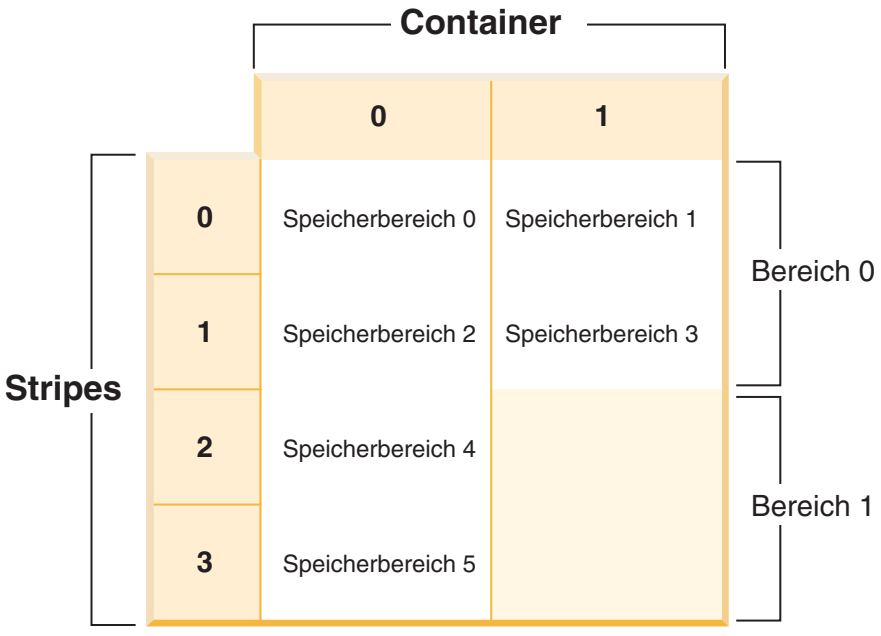


Abbildung 13. Tabellenbereich mit zwei Containern, Bereiche hervorgehoben

Stripes 0 und 1 enthalten beide Container (0 und 1), jedoch enthalten die Stripes 2 und 3 nur den ersten Container (0). Jede dieser Gruppen von Stripes wird als Be-

reich (range) bezeichnet. Die Tabellenbereichszuordnung, die in der Momentaufnahme eines Tabellenbereichs gezeigt wird, sieht folgendermaßen aus:

```
Range   Stripe Stripe   Max      Max   Start   End    Adj.  Containers
Number  Set    Offset   Extent   Page  Stripe  Stripe
    [0]    [0]      0        3     99       0       1  0    2 (0, 1)
    [1]    [0]      0        5    149       2       3  0    1 (0)
```

Im ersten Bereich (Range) befinden sich vier EXTENTSIZE große Speicherbereiche (Extents). Daher ist 3 die höchste Speicherbereichsnummer (Max Extent), die in diesem Bereich adressiert wird. Jeder Speicherbereich ist 25 Seiten groß, sodass sich im ersten Bereich 100 Seiten befinden. Da die Seitennummerierung ebenfalls bei 0 beginnt, ist 99 die höchste Seitennummer (Max Page), die in diesem Bereich adressiert wird. Der erste Stripe (Start Stripe) in diesem Bereich ist 0 und der letzte Stripe (End Stripe) im Bereich ist Stripe 1. Es gibt zwei Container in diesem Bereich, 0 und 1. Das Stripe-Offset ist der erste Stripe im Stripe-Set, in diesem Fall 0, weil nur ein Stripe-Set vorhanden ist. Die Bereichsanpassung (Adj.) ist ein Offsetwert, der verwendet wird, wenn Daten in einem Tabellenbereich neu verteilt werden. (Eine Neuverteilung kann stattfinden, wenn in einem Tabellenbereich Speicherplatz hinzugefügt oder gelöscht wird.) Wenn keine Neuverteilung stattfindet, ist dieser Wert immer 0.

Im zweiten Bereich (Range) befinden sich zwei EXTENTSIZE große Speicherbereiche (Extents), und da 3 die höchste Speicherbereichsnummer ist, die im vorigen Bereich adressiert wurde, ist 5 die höchste Speicherbereichsnummer, die in diesem Bereich adressiert wird. Im zweiten Bereich befinden sich 50 Seiten (2 Speicherbereiche * 25 Seiten), und da 99 die höchste Seitennummer ist, die im vorigen Bereich adressiert wird, ist 149 nun die höchste Seitennummer, die in diesem Bereich adressiert wird. Dieser Bereich beginnt bei Stripe 3 und endet bei Stripe 3.

Automatische Änderung der Größe von DMS-Tabellenbereichen:

Durch die Aktivierung der Funktion zur automatischen Größenänderung für DMS-Tabellenbereiche (DMS, vom Datenbankmanager verwalteter Tabellenbereich), die Dateicontainer verwenden, kann der Datenbankmanager die Problembedingung eines vollen Tabellenbereichs automatisch durch Erweitern vorhandener Container beheben.

DMS-Tabellenbereiche bestehen aus Dateicontainern oder Containern für (unformatierte und unpartitionierte) Roheinheiten. Ihre Größe wird definiert, wenn die Container dem Tabellenbereich zugeordnet werden. Der Tabellenbereich gilt dann als voll, wenn der gesamte Speicherplatz des Containers belegt ist. Im Gegensatz zu SMS-Tabellenbereichen können hier jedoch Container mit der Anweisung ALTER TABLESPACE manuell hinzugefügt oder erweitert werden, wodurch der Tabellenbereich mehr Speicherplatz erhält. DMS-Tabellenbereiche verfügen darüber hinaus über eine *Funktion zur automatischen Größenänderung (AUTORESIZE)*: Wenn in einem DMS-Tabellenbereich, für den die automatische Größenänderung aktiviert ist, der Speicherplatz erschöpft ist, vergrößert der Datenbankmanager den Tabellenbereich, indem er einen oder mehrere Dateicontainer erweitert.

Die Funktionalität der automatischen Größenänderung für DMS-Tabellenbereiche hängt mit der Funktionalität von Tabellenbereichen mit dynamischem Speicher zusammen, ist jedoch nicht dasselbe. Weitere Informationen finden Sie in „Vergleich von SMS-Tabellenbereichen, DMS-Tabellenbereichen und Tabellenbereichen mit dynamischem Speicher” auf Seite 201.

Aktivieren und Inaktivieren der Funktion zur automatischen Größenänderung (AUTORESIZE)

Standardmäßig ist die Funktion zur automatischen Größenänderung für einen DMS-Tabellenbereich nicht aktiviert. Mit der folgenden Anweisung wird ein DMS-Tabellenbereich ohne Aktivierung der automatischen Größenänderung erstellt:

```
CREATE TABLESPACE DMS1 MANAGED BY DATABASE
  USING (FILE '/db2files/DMS1' 10 M)
```

Zum Aktivieren der Funktion zur automatischen Größenänderung geben Sie die Klausel AUTORESIZE YES in der Anweisung CREATE TABLESPACE an:

```
CREATE TABLESPACE DMS1 MANAGED BY DATABASE
  USING (FILE '/db2files/DMS1' 10 M) AUTORESIZE YES
```

Sie können die Funktion zur automatischen Größenänderung auch nach der Erstellung eines DMS-Tabellenbereichs aktivieren oder inaktivieren, indem Sie die Klausel AUTORESIZE in der Anweisung ALTER TABLESPACE verwenden:

```
ALTER TABLESPACE DMS1 AUTORESIZE YES
ALTER TABLESPACE DMS1 AUTORESIZE NO
```

Darüber hinaus sind zwei weitere Attribute, MAXSIZE und INCREASESIZE, für die automatische Größenänderung von Tabellenbereichen relevant:

Maximalgröße (MAXSIZE)

Mit der Klausel MAXSIZE der Anweisung CREATE TABLESPACE wird die maximal mögliche Größe für den Tabellenbereich definiert. So wird beispielsweise durch die folgende Anweisung ein Tabellenbereich erstellt, der bis zu einem Maximalwert von 100 MB vergrößert werden kann (pro Datenbankpartition, falls die Datenbank mehrere Datenbankpartitionen besitzt):

```
CREATE TABLESPACE DMS1 MANAGED BY DATABASE
  USING (FILE '/db2files/DMS1' 10 M)
  AUTORESIZE YES MAXSIZE 100 M
```

Die Klausel MAXSIZE NONE gibt an, dass für den Tabellenbereich kein Maximalwert festgelegt ist. Der Tabellenbereich kann vergrößert werden, bis ein Dateisystemgrenzwert oder der Tabellenbereichsgrenzwert erreicht wird (siehe „SQL- und XML-Begrenzungen" (SQL and XML limits) im Handbuch *SQL Reference*). Wenn Sie die Klausel MAXSIZE nicht angeben, gibt es keine Maximalgröße, wenn die Funktion zur automatischen Größenänderung aktiviert wird.

Mit der Anweisung ALTER TABLESPACE können Sie den Wert von MAXSIZE für einen Tabellenbereich ändern, für den die Funktion zur automatischen Größenänderung bereits aktiviert ist, wie in den folgenden Beispielen gezeigt:

```
ALTER TABLESPACE DMS1 MAXSIZE 1 G
ALTER TABLESPACE DMS1 MAXSIZE NONE
```

Wenn Sie eine Maximalgröße angeben, kann der tatsächlich vom Datenbankmanager umgesetzte Wert geringfügig niedriger sein als der angegebene Wert, da der Datenbankmanager versucht, die Vergrößerung von Containern konsistent zu halten.

Vergrößerungsvolumen (INCREASESIZE)

Mit der Klausel INCREASESIZE der Anweisung CREATE TABLESPACE wird die Menge an Speicherplatz definiert, um die der Tabellenbereich vergrößert wird,

wenn keine freien Speicherbereiche im Tabellenbereich mehr vorhanden sind, jedoch eine Anforderung für einen oder mehrere Speicherbereiche erfolgt ist. Sie können den Wert als explizite Größe oder als Prozentsatz wie in den folgenden Beispielen angeben:

```
CREATE TABLESPACE DMS1 MANAGED BY DATABASE
  USING (FILE '/db2files/DMS1' 10 M)
  AUTORESIZE YES INCREASESIZE 5 M

CREATE TABLESPACE DMS1 MANAGED BY DATABASE
  USING (FILE '/db2files/DMS1' 10 M)
  AUTORESIZE YES INCREASESIZE 50 PERCENT
```

Ein Prozentsatz bedeutet, dass das Volumen, um das zu vergrößern ist, jedes Mal, wenn der Tabellenbereich vergrößert werden muss, neu berechnet wird. Dabei basiert das jeweilige Vergrößerungsvolumen auf einem Prozentsatz der Tabellenbereichsgröße zu dem betreffenden Zeitpunkt. Beispiel: Wenn der Tabellenbereich 20 MB groß ist und für INCREASESIZE ein Wert von 50 % angegeben wird, wird der Tabellenbereich bei der ersten Vergrößerung um 10 MB erweitert (auf eine Größe von 30 MB) und bei der nächsten Vergrößerung um 15 MB.

Wenn Sie die Klausel INCREASESIZE bei der Aktivierung der Funktion zur automatischen Größenänderung nicht angegeben, ermittelt der Datenbankmanager einen angemessenen Wert, der sich während des Lebenszyklus des Tabellenbereichs ändern kann. Wie bei AUTORESIZE und MAXSIZE können Sie den Wert für INCREASESIZE mithilfe der Anweisung ALTER TABLESPACE ändern.

Wenn Sie einen Wert für INCREASESIZE angeben, kann der tatsächlich vom Datenbankmanager verwendete Wert geringfügig vom angegebenen Wert abweichen. Diese Anpassung des verwendeten Wertes dient dazu, die Vergrößerungen für alle Container im Tabellenbereich konsistent zu halten.

Einschränkungen für die Verwendung von AUTORESIZE mit DMS-Tabellenbereichen

- Sie können diese Funktion nicht für Tabellenbereiche verwenden, die mit Containern für Roheinheiten arbeiten, und Sie können einem Tabellenbereich, für den die automatische Größenänderung aktiviert ist, keine Container für Roheinheiten hinzufügen. Versuche, solche Operationen auszuführen, schlagen mit Fehlern (SQL0109N) fehl. Wenn Sie Container für Roheinheiten hinzufügen müssen, müssen Sie die Funktion zur automatischen Größenänderung zuvor inaktivieren.
- Wenn Sie die Funktion zur automatischen Größenänderung inaktivieren, werden die Werte, die INCREASESIZE und MAXSIZE zugeordnet sind, nicht beibehalten und stehen bei einer späteren Aktivierung der Funktion nicht mehr zur Verfügung.
- Eine umgeleitete Restoreoperation kann die Containerdefinitionen nicht so ändern, dass ein Container für eine Roheinheit verwendet werden kann. Ein Versuch, diese Art von Operation auszuführen, verursacht einen Fehler (SQL0109N).
- Neben der Begrenzung des Volumens, um das der Datenbankmanager einen Tabellenbereich automatisch vergrößert, begrenzt die Maximalgröße auch das Ausmaß, bis zu dem Sie einen Tabellenbereich manuell vergrößern können. Wenn Sie eine Operation ausführen, durch die einem Tabellenbereich Speicherbereich hinzugefügt wird, muss die resultierende Größe kleiner oder gleich der Maximalgröße sein. Speicherbereich kann mit der Klausel ADD, EXTEND, RESIZE oder BEGIN NEW STRIPE SET der Anweisung ALTER TABLESPACE hinzugefügt werden.

Erweiterung von Tabellenbereichen

Wenn AUTORESIZE aktiviert ist, versucht der Datenbankmanager, den Tabellenbereich zu vergrößern, wenn der gesamte vorhandene Speicherbereich belegt ist und eine Anforderung für zusätzlichen Speicherbereich erfolgt. Der Datenbankmanager bestimmt, welche Container im Tabellenbereich erweitert werden können, sodass keine Neuverteilung der Daten im Tabellenbereich stattfindet. Der Datenbankmanager erweitert nur die Container, die sich im letzten Bereich der Tabellenbereichszuordnung befinden (diese Zuordnung beschreibt die Speicherbelegung für den Tabellenbereich; weitere Informationen siehe „Tabellenbereichszuordnungen für DMS-Tabellenbereiche” auf Seite 176), und erweitert sie um dasselbe Volumen.

Betrachten Sie zum Beispiel die folgende Anweisung:

```
CREATE TABLESPACE TS1 MANAGED BY DATABASE
  USING (FILE 'C:\TS1CONT' 1000, FILE 'D:\TS1CONT' 1000,
         FILE 'E:\TS1CONT' 2000, FILE 'F:\TS1CONT' 2000)
  EXTENTSIZE 4
  AUTORESIZE YES
```

Unter Berücksichtigung der Tatsache, dass der Datenbankmanager einen kleinen Teil jedes Containers (einen EXTENTSIZE großen Speicherbereich) für Metadaten verwendet, ist nachfolgend die Tabellenbereichszuordnung dargestellt, die für den Tabellenbereich auf der Basis der Anweisung CREATE TABLESPACE erstellt wird. (Die Tabellenbereichszuordnung ist Teil der Ausgabe einer Momentaufnahme für einen Tabellenbereich.)

```
Tabellenbereichszuordnung:

 Range  Stripe Stripe  Max         Max  Start  End    Adj.   Containers
 Number Set    Offset  Extent      Page Stripe Stripe
 [   0] [   0]      0     995      3983      0    248   0    4 (0,1,2,3)
 [   1] [   0]      0    1495      5983    249    498   0    2 (2,3)
```

Die Tabellenbereichszuordnung zeigt, dass die Container mit der ID 2 und 3 (`E:\TS1CONT` und `F:\TS1CONT`) die einzigen Container im letzten Bereich (Range) der Zuordnung sind. Wenn der Datenbankmanager die Container in diesem Tabellenbereich automatisch erweitert, werden daher nur diese beiden Container erweitert.

Anmerkung: Wenn Sie einen Tabellenbereich erstellen, bei dem alle Container dieselbe Größe haben, enthält die Zuordnung nur einen Bereich (Range). In diesem Fall erweitert der Datenbankmanager jeden Container. Um zu verhindern, dass die Erweiterung nur auf eine Untermenge der Container begrenzt wird, müssen Sie einen Tabellenbereich mit Containern gleicher Größe erstellen.

Wie zuvor erläutert, können Sie eine Begrenzung für die Maximalgröße des Tabellenbereichs angeben. Sie können auch den Wert NONE angeben, sodass keine Größenbegrenzung besteht. Wenn Sie NONE oder keinen Grenzwert angeben, wird die Obergrenze durch den Grenzwert des Dateisystems oder des Tabellenbereichs definiert. Der Datenbankmanager versucht nicht, die Tabellenbereichsgröße über die Obergrenze hinaus zu erweitern. Es kann jedoch vorkommen, dass ein Versuch, einen Container zu erweitern, fehlschlägt, bevor dieser Grenzwert erreicht ist, wenn das Dateisystem voll ist. In diesem Fall vergrößert der Datenbankmanager den Tabellenbereich nicht weiter und gibt eine Fehlerbedingung wegen nicht ausreichenden Speicherplatzes an die Anwendung zurück. In dieser Situation haben Sie zwei Möglichkeiten zur Lösung des Problems:

- Stellen Sie mehr verfügbaren Speicherplatz in dem vollen Dateisystem bereit.
- Führen Sie Containeroperationen für den Tabellenbereich aus, die bewirken, dass sich der betreffende Container nicht mehr im letzten Bereich (Range) der Tabel-

lenbereichszuordnung befindet. Die einfachste Methode, dies zu erreichen, besteht darin, dem Tabellenbereich ein neues Stripe-Set mit einer neuen Gruppe von Containern hinzuzufügen, wobei nach Möglichkeit sichergestellt werden sollte, dass alle Container dieselbe Größe haben. Sie können neue Stripe-Sets mit der Klausel BEGIN NEW STRIPE SET in der Anweisung ALTER TABLESPACE hinzufügen. Durch das Hinzufügen eines neuen Stripe-Sets wird der Tabellenbereichszuordnung ein neuer Bereich hinzugefügt. Durch einen neu hinzugefügten Bereich befinden sich die Container, die der Datenbankmanager automatisch zu erweitern versucht, in diesem neuen Stripe-Set, und die älteren Container bleiben unverändert.

Anmerkung: Wenn eine vom Benutzer eingeleitete Containeroperation ansteht oder gerade eine nachfolgende Neuverteilung ausgeführt wird, wird die Funktion zur automatischen Größenänderung so lange inaktiviert, bis die Operation festgeschrieben bzw. die Neuverteilung abgeschlossen ist.

Ein Beispiel für DMS-Tabellenbereiche: Nehmen Sie einmal an, dass ein Tabellenbereich drei Container umfasst, die dieselbe Größe haben und sich jeweils in einem eigenen Dateisystem befinden. Während der Ausführung von Operationen im Tabellenbereich erweitert der Datenbankmanager diese drei Container automatisch. Schließlich ist eines der Dateisysteme voll, und der zugehörige Container kann nicht mehr erweitert werden. Wenn es nicht möglich ist, im Dateisystem zusätzlichen freien Speicherplatz verfügbar zu machen, müssen Sie Containeroperationen im Tabellenbereich durchführen, die bewirken, dass sich der betreffende Container nicht mehr im letzten Bereich (Range) der Tabellenbereichszuordnung befindet. In diesem Fall könnten Sie beispielsweise ein neues Stripe-Set hinzufügen und dabei zwei Container angeben (jeweils einen für jedes Dateisystem, in dem noch Speicherplatz zur Verfügung steht); oder Sie könnten eine höhere Anzahl von Containern angeben (und dabei sicherstellen, dass jeder der hinzugefügten Container dieselbe Größe hat und in jedem der verwendeten Dateisysteme ausreichend Speicherplatz für mögliche Erweiterungen verfügbar ist). Beim Vergrößern des Tabellenbereichs versucht der Datenbankmanager nun, anstelle der älteren Container die Container in diesem neuen Stripe-Set zu erweitern.

Überwachung

Informationen zur automatischen Größenänderung von DMS-Tabellenbereichen sind Teil der Momentaufnahmenausgabe des Tabellenbereichsmonitors. Die Werte für das Vergrößerungsvolumen (INCREASESIZE) und für die Maximalgröße (MAXSIZE) werden ebenfalls in der Ausgabe wie im folgenden Beispiel angezeigt:

```
Autom. Größenänderung aktiviert               = Ja oder Nein
Aktuelle Größe des Tabellenbereichs (Byte)    = ###
Maximale Größe des Tabellenbereichs (Byte)    = ### oder NONE
Vergrößerungsvolumen (Byte)                   = ###
Vergrößerungsvolumen (Prozent)                = ###
Zeit der letzten erfolgreichen Größenänderung = TT/MM/JJJJ HH:MM:SS.SSSSSS
Letzte fehlgeschlagene Größenänderung         = Ja oder Nein
```

Tabellenbereiche mit dynamischem Speicher

Bei Verwendung von Tabellenbereichen mit dynamischem Speicher erfolgt die Speicherverwaltung automatisch. Der Datenbankmanager erstellt und erweitert Container nach Bedarf.

Anmerkung: Obwohl Sie beim Erstellen einer Datenbank die Klausel AUTOMATIC STORAGE NO angeben können, ist die Klausel AUTOMATIC STORAGE veraltet und wird in einem zukünftigen Release möglicherweise entfernt.

Alle Tabellenbereiche, die Sie erstellen, werden als Tabellenbereiche mit dynamischem Speicher verwaltet, sofern Sie nichts anderes angeben oder die Datenbank unter Verwendung der Klausel AUTOMATIC STORAGE NO erstellt wurde. Bei Tabellenbereichen mit dynamischem Speicher brauchen Sie keine Containerdefinitionen anzugeben. Der Datenbankmanager sorgt für die Erstellung und Erweiterung von Containern, um den Speicher zu nutzen, der der Datenbank zugeordnet ist. Wenn Sie einer Speichergruppe Speicher hinzufügen, werden automatisch neue Container erstellt, wenn die vorhandenen Container ihre maximale Kapazität erreichen. Wenn Sie den neu hinzugefügten Speicher unverzüglich nutzen wollen, können Sie eine Neuverteilung der Daten des Tabellenbereichs durchführen, wobei die Daten über die neue, erweiterte Gruppe von Containern und Stripe-Sets neu zugeordnet werden. Falls Sie jedoch weniger an E/A-Parallelität interessiert sind und Ihrem Tabellenbereich nur Kapazität hinzufügen wollen, können Sie die Neuverteilung unterlassen. In diesem Fall werden neue Stripe-Sets erstellt, wenn neuer Speicherplatz benötigt wird.

Tabellenbereiche mit dynamischem Speicher können in einer Datenbank mithilfe der Anweisung CREATE TABLESPACE erstellt werden. Neue Tabellenbereiche in einer Datenbank sind standardmäßig Tabellenbereiche mit dynamischem Speicher, d. h. die Klausel MANAGED BY AUTOMATIC STORAGE ist optional. Sie können beim Erstellen eines Tabellenbereichs mit dynamischem Speicher auch Optionen angeben. Dazu gehören zum Beispiel die Anfangsgröße, der Betrag, um den der Tabellenbereich vergrößert wird, wenn er voll ist, die Maximalgröße, auf die der Tabellenbereich anwachsen kann und die von dem Tabellenbereich verwendete Speichergruppe. Die folgenden Beispiele zeigen einige Anweisungen zur Erstellung von Tabellenbereichen mit dynamischem Speicher:

```
CREATE TABLESPACE TS1
CREATE TABLESPACE TS2 MANAGED BY AUTOMATIC STORAGE
CREATE TEMPORARY TABLESPACE TEMPTS
CREATE USER TEMPORARY TABLESPACE USRTMP MANAGED BY AUTOMATIC STORAGE
CREATE LARGE TABLESPACE LONGTS
CREATE TABLESPACE TS3 INITIALSIZE 8K INCREASESIZE 20 PERCENT MANAGED BY AUTOMATIC STORAGE
CREATE TABLESPACE TS4 MAXSIZE 2G   CREATE TABLESPACE TS5 USING STOGROUP SG_HOT
```

In jedem dieser Beispiele wird angenommen, dass die Datenbank, für die diese Tabellenbereiche erstellt werden, über mindestens eine definierte Speichergruppe verfügt. Wenn Sie einen Tabellenbereich in einer Datenbank erstellen, für die keine Speichergruppen definiert sind, können Sie die Klausel MANAGED BY AUTOMATIC STORAGE nicht verwenden. Stattdessen müssen Sie eine der folgenden Aktionen ausführen:

- Geben Sie die Klausel MANAGED BY SYSTEM oder MANAGED BY DATABASE der Anweisung CREATE TABLESPACE an. Durch die Verwendung dieser Klauseln wird ein vom System verwalteter SMS-Tabellenbereich bzw. ein vom Datenbankmanager verwalteter Tabellenbereich (DMS-Tabellenbereich) erstellt. In beiden Fällen müssen Sie eine explizite Liste von Containern angeben.
- Erstellen Sie eine Speichergruppe und versuchen Sie anschließend erneut, einen Tabellenbereich mit dynamischem Speicher zu erstellen.

Verwaltung der Speichererweiterung bei Tabellenbereichen mit dynamischem Speicher:

Wenn Sie *Tabellenbereiche mit dynamischem Speicher* verwenden, sorgt der Datenbankmanager nach Bedarf für die Erstellung und Erweiterung von Containern. Wenn Sie Speicher zu der vom Tabellenbereich verwendeten Speichergruppe hinzufügen, werden neue Container automatisch erstellt. Wie der neue Speicherbereich genutzt wird, hängt jedoch davon ab, ob Sie eine Neuverteilung der Daten (REBALANCE) für den Tabellenbereich ausführen oder nicht.

Wenn ein Tabellenbereich mit dynamischem Speicher erstellt wird, erstellt der Datenbankmanager einen Container in jedem der Speicherpfade der Speichergruppe, deren Verwendung definiert ist (sofern der Speicherplatz dies zulässt). Wenn sämtlicher Speicherplatz in einem Tabellenbereich belegt ist, erhöht der Datenbankmanager automatisch die Größe des Tabellenbereichs, indem er Container erweitert oder ein neues Stripe-Set von Containern hinzufügt.

Der Speicher für Tabellenbereiche mit dynamischem Speicher wird auf der Datenbankebene verwaltet. Das heißt, Sie fügen nicht Tabellenbereichen (wie bei DMS-Tabellenbereichen), sondern den *Speichergruppen* der Datenbank Speicher hinzu. Wenn Sie einer von dem Tabellenbereich verwendeten Speichergruppe Speicher hinzufügen, erstellt die Funktion für dynamischen Speicher je nach Bedarf neue Container zum Speichern der Daten. Tabellenbereiche, die bereits vorhanden sind, beginnen jedoch nicht sofort damit, Speicher in den neuen Pfaden zu belegen. Wenn ein Tabellenbereich wachsen muss, versucht der Datenbankmanager zunächst, die Container im letzten *Bereich* ('Range') des Tabellenbereichs zu erweitern. Unter einem Bereich sind alle Container über ein bestimmtes Stripe-Set zu verstehen. Wenn dies erfolgreich ist, beginnen Anwendungen damit, diesen neuen Speicher zu verwenden. Wenn jedoch der Versuch, die Container zu erweitern fehlschlägt, wie dies zum Beispiel geschehen kann, wenn eines oder mehrere Dateisysteme voll sind, versucht der Datenbankmanager ein neues Stripe-Set von Containern zu erstellen. Erst an diesem Punkt zieht der Datenbankmanager in Betracht, die neu hinzugefügten Speicherpfade für den Tabellenbereich zu verwenden. Abb. 14 auf Seite 188 veranschaulicht diesen Prozess.

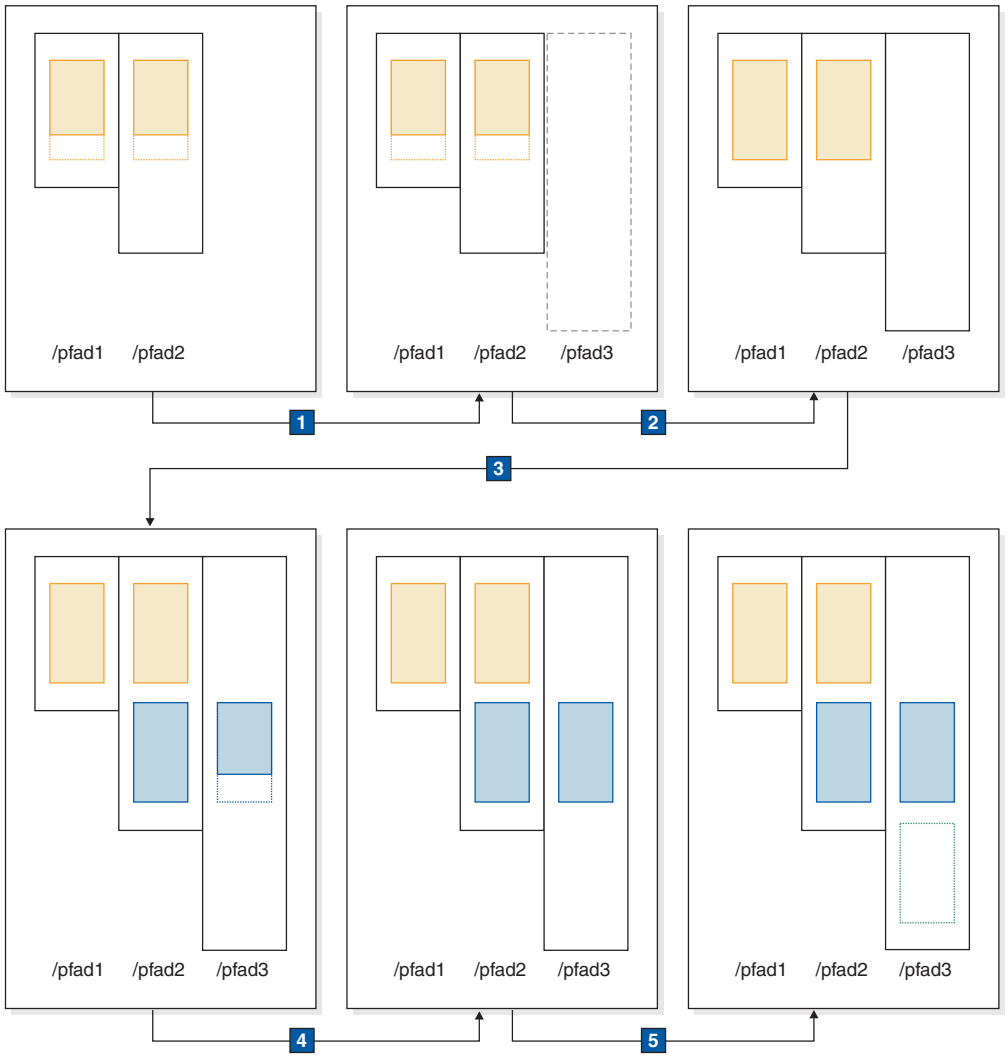


Abbildung 14. Bedarfsgesteuertes Hinzufügen von Containern durch die Funktion für dynamischen Speicher

Das oben gezeigte Diagramm beschreibt Folgendes:

1. Der Tabellenbereich hat zu Anfang zwei Container, die noch nicht bis zur maximalen Kapazität gefüllt sind. Der Speichergruppe wird ein neuer Speicherpfad mithilfe der Anweisung ALTER STOGROUP mit der Klausel ADD hinzugefügt. Der neue Speicherpfad wird jedoch vorerst nicht verwendet.
2. Die Belegung der beiden ursprünglichen Container erreicht die maximale Kapazität.
3. Ein neues Stripe-Set von Containern hinzugefügt und die Container beginnen sich mit Daten zu füllen.
4. Die Belegung der Container im neuen Stripe-Set erreicht die maximale Kapazität der Container.
5. Ein neues Stripe-Set wird hinzugefügt, weil kein Platz für ein weiteres Anwachsen der Container vorhanden ist.

Wenn der Tabellenbereich mit dynamischem Speicher unverzüglich mit der Verwendung des neu hinzugefügten Speicherpfads beginnen soll, können Sie eine Neuverteilung mithilfe der Klausel REBALANCE des Befehls ALTER TABLESPACE ausführen. Wenn Sie die Daten Ihres Tabellenbereichs neu verteilen, werden sie

den Containern und Stripe-Sets im neu hinzugefügten Speicher neu zugeordnet. Dies ist in Abb. 15 dargestellt.

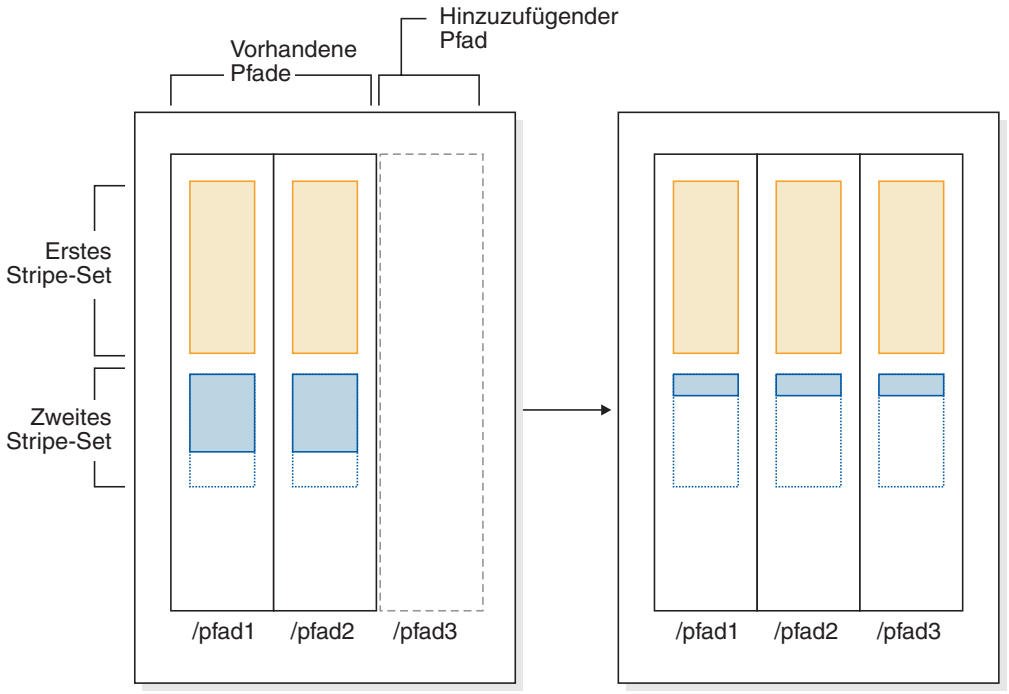


Abbildung 15. Ergebnisse des Hinzufügens neuen Speichers und einer Neuverteilung der Daten des Tabellenbereichs

In diesem Beispiel wird nicht ein neues Stripe-Set erstellt, sondern der Neuverteilungsprozess erweitert die vorhandenen Stripe-Sets auf den neuen Speicherpfad, indem er nach Bedarf Container erstellt, und anschließend die Daten über alle Container neu zuordnet.

Containernamen in Tabellenbereichen mit dynamischem Speicher:

Obwohl Containernamen für Tabellenbereiche mit dynamischem Speicher durch den Datenbankmanager zugeordnet werden, sind sie sichtbar, wenn Sie Befehle wie **LIST TABLESPACE CONTAINERS** oder **GET SNAPSHOT FOR TABLESPACES** ausführen. In diesem Abschnitt werden die Konventionen für Containernamen beschrieben, sodass Sie sie erkennen können, wenn sie angezeigt werden.

Die Namen, die Containern in Tabellenbereichen mit dynamischem Speicher zugeordnet werden, sind wie folgt strukturiert:

```
speicherpfad/instanzname/NODE####/datenbankname/T#######/C#######.ERW
```

Dabei gilt:

speicherpfad
: Ein Speicherpfad, der einer Speichergruppe zugeordnet wurde

instanzname
: Die Instanz, unter der die Datenbank erstellt wurde

datenbankname
: Der Name der Datenbank

NODE####
Die Datenbankpartitionsnummer (z. B. NODE0000)

T#######
Die Tabellenbereichs-ID (z. B. T0000003)

C#######
Die Container-ID (z. B. C0000012)

ERW Eine Erweiterung basierend auf dem Typ der zu speichernden Daten:

CAT Systemkatalogtabellenbereich

TMP Tabellenbereich für temporäre Systemtabellen

UTM Tabellenbereich für temporäre Benutzertabellen

USR Benutzertabellenbereich oder regulärer Tabellenbereich

LRG LOB-Tabellenbereich

Beispiel

Nehmen Sie zum Beispiel an, dass ein Tabellenbereich mit dynamischem Speicher mit dem Namen TBSAUTO in der Datenbank SAMPLE erstellt wurde. Wenn der Befehl LIST TABLESPACES ausgeführt wird, wird er mit der Tabellenbereichs-ID **10** angezeigt:

```
 Tabellenbereichs-ID                  = 10
 Name                                 = TBSAUTO
 Typ                                  = Von der Datenbank verwalteter Bereich
 Inhalt                               = Alle permanenten Daten. Großer Tabellenbereich.
 Status                               = 0x0000
   Detaillierte Erläuterung:
     Normal
```

Wenn Sie den Befehl **LIST TABLESPACE CONTAINERS** für den Tabellenbereich mit der ID 10 jetzt ausführen, sehen Sie die den Containern für diesen Tabellenbereich zugeordneten Namen:

```
LIST TABLESPACE CONTAINERS FOR 10 SHOW DETAIL

            Tabellenbereichscontainer für Tabellenbereich 10

 Container-ID                         = 0
 Name                                 = D:\DB2\NODE0000\SAMPLE\T0000010\C0000000.LRG
 Typ                                  = Datei
 Seiten insgesamt                     = 4096
 Verwendbare Seiten                   = 4064
 Im Zugriff                           = Ja
```

In diesem Beispiel können Sie den Namen des Containers (Container-ID = 0) für diesen Tabellenbereich sehen:

```
D:\DB2\NODE0000\SAMPLE\T0000010\C0000000.LRG
```

Konvertieren von Tabellenbereichen zur Verwendung von dynamischem Speicher:

Sie können einige oder alle DMS-Tabellenbereiche (DMS, Database Managed Space, vom Datenbankmanager verwalteter Tabellenbereich) in einer Datenbank konvertieren, sodass dynamischer Speicher verwendet wird. Die Verwendung von dynamischem Speicher vereinfacht Ihnen die Aufgaben des Speichermanagements.

Vorbereitende Schritte

Stellen Sie sicher, dass die Datenbank über mindestens eine Speichergruppe verfügt. Führen Sie dazu eine Abfrage für SYSCAT.STOGROUPS aus und geben Sie die Anweisung CREATE STOGROUP ein, wenn die Ergebnismenge leer ist.

Vorgehensweise

Zur Konvertierung eines DMS-Tabellenbereichs zur Verwendung von dynamischem Speicher wählen Sie eine der drei folgenden Methoden aus:

- **Ändern Sie einen einzelnen Tabellenbereich.** Bei dieser Methode bleibt der Tabellenbereich online. Allerdings ist eine Neuverteilungsoperation erforderlich, die einige Zeit benötigt, um Daten aus den Containern ohne dynamischen Speicher in die neuen Container mit dynamischem Speicher zu versetzen.
  1. Geben Sie den Tabellenbereich an, den Sie zur Verwendung von dynamischem Speicher konvertieren möchten. Geben Sie an, welche Speichergruppe der Tabellenbereich verwenden soll. Geben Sie die folgende Anweisung ein:

     ```
     ALTER TABLESPACE tabber1 MANAGED BY AUTOMATIC STORAGE USING STOGROUP sg-medium
     ```

     Dabei ist `tabber1` der Tabellenbereich und `sg-medium` die Speichergruppe, für die der Tabellenbereich definiert ist.
  2. Versetzen Sie die benutzerdefinierten Daten aus den alten Containern in die Speicherpfade der Speichergruppe `sg_medium`. Geben Sie dazu die folgende Anweisung ein:

     ```
     ALTER TABLESPACE tabber1 REBALANCE
     ```

     Anmerkung: Wenn Sie die Option REBALANCE jetzt nicht angeben und die Anweisung ALTER TABLESPACE später mit der Option REDUCE eingeben, werden die Container mit dynamischem Speicher entfernt. Um dieses Problem zu vermeiden, geben Sie die Anweisung ALTER TABLESPACE unter Angabe der Option REBALANCE ein.
  3. Verwenden Sie die folgende Anweisung, um den Fortschritt der REBALANCE-Operation zu überwachen:

     ```
     SELECT * from tabelle (MON_GET_REBALANCE_STATUS( 'tabber1', -2))
     ```

- **Verwenden Sie eine umgeleitete Restoreoperation.** Wenn die umgeleitete Restoreoperation in Bearbeitung ist, können Sie auf die momentan konvertierten Tabellenbereiche nicht zugreifen. Bei einem vollständigen umgeleiteten Restore für eine Datenbank besteht auf keinen der Tabellenbereiche Zugriff, bis die Recovery abgeschlossen ist.
  1. Führen Sie den Befehl **RESTORE DATABASE** aus, indem Sie den Parameter **REDIRECT** angeben. Wenn Sie einen einzelnen Tabellenbereich konvertieren wollen, geben Sie außerdem den Parameter **TABLESPACE** an:

     ```
     RESTORE DATABASE datenbankname TABLESPACE (tabellenbereichsname) REDIRECT
     ```
  2. Führen Sie den Befehl **SET TABLESPACE CONTAINERS** unter Angabe des Parameters **USING AUTOMATIC STORAGE** für jeden Tabellenbereich aus, den Sie konvertieren wollen:

     ```
     SET TABLESPACE CONTAINERS FOR tabellenbereichs-id USING AUTOMATIC STORAGE
     ```
  3. Führen Sie den Befehl **RESTORE DATABASE** erneut aus, indem Sie diesmal den Parameter **CONTINUE** angeben:

     ```
     RESTORE DATABASE datenbankname CONTINUE
     ```
  4. Führen Sie den Befehl **ROLLFORWARD DATABASE** aus, indem Sie die Parameter **TO END OF LOGS** und **AND STOP** angeben:

     ```
     ROLLFORWARD DATABASE datenbankname TO END OF LOGS AND STOP
     ```

Bei Verwendung einer umgeleiteten Restoreoperation muss eine zusätzliche Anweisung ALTER TABLESPACE eingegeben werden, um die Datenbankkataloge mit der korrekten Speichergruppenzuordnung für den Tabellenbereich zu aktualisieren. Die Zuordnung zwischen Tabellenbereichen und Speichergruppen wird in den Systemkatalogtabellen aufgezeichnet und während der umgeleiteten Restoreoperation nicht aktualisiert. Die Anweisung ALTER TABLESPACE aktualisiert nur die Katalogtabellen; die zusätzliche Verarbeitungszeit für eine REBALANCE-Operation entfällt. Wenn die Anweisung ALTER TABLESPACE nicht eingegeben wird, kann sich dies auf die Abfrageleistung auswirken. Wenn Sie die Standardspeichergruppe für den Tabellenbereich während der umgeleiteten Restoreoperation geändert haben, müssen Sie den Befehl **RESTORE DATABASE** mit dem Parameter **USING STOGROUP** eingeben, um die Konsistenz sämtlicher Datenbankpartitionen und Systemkataloge zu gewährleisten.

Beispiel

Gehen Sie wie folgt vor, um den vom Datenbankmanager verwalteten Tabellenbereich *SALES* während einer umgeleiteten Restoreoperation zur Verwendung von dynamischem Speicher zu konvertieren:

1. Zum Konfigurieren einer umgeleiteten Restoreoperation für `testdb` geben Sie den folgenden Befehl ein:
   ```
   RESTORE DATABASE testdb REDIRECT
   ```
2. Geben Sie an, dass der Tabellenbereich *SALES* mit dynamischem Speicher verwaltet werden soll. Der Tabellenbereich *SALES* verfügt über den ID-Wert 5.
   ```
   SET TABLESPACE CONTAINERS FOR 5 USING AUTOMATIC STORAGE
   ```

   Anmerkung: Zur Bestimmung des ID-Werts eines Tabellenbereichs während einer umgeleiteten Restoreoperation verwenden Sie die Option GENERATE SCRIPT des Befehls **RESTORE DATABASE**.
3. Setzen Sie den folgenden Befehl ab, um mit der Restoreoperation fortzufahren:
   ```
   RESTORE DATABASE testdb CONTINUE
   ```
4. Aktualisieren Sie die Speichergruppeninformationen in den Katalogtabellen.
   ```
   CONNECT TO testdb
   ALTER TABLESPACE SALES MANAGED BY AUTOMATIC STORAGE
   ```
5. Wenn Sie die Speichergruppe für den Tabellenbereich während der umgeleiteten Restoreoperation geändert haben, geben Sie den folgenden Befehl ein:
   ```
   RESTORE DATABASE testdb USING STOGROUP sg_standard
   ```

Obere Grenze für Tabellenbereiche

Die Bezeichnung *obere Grenze* gibt die Seitenzahl der ersten Seite in dem Speicherbereich an, der auf den letzten zugeordneten Speicherbereich folgt.

Wenn zum Beispiel ein Tabellenbereich 1000 Seiten enthält und EXTENTSIZE den Wert 10 hat, sind 100 Speicherbereiche (der Größe EXTENTSIZE) vorhanden. Wenn der 42ste Speicherbereich der höchste zugeordnete Speicherbereich im Tabellenbereich ist, bedeutet dies, dass die obere Grenze den Wert 420 hat.

Tipp: Speicherbereiche werden von 0 an indexiert. Die obere Grenze ist dementsprechend die *letzte Seite des höchsten zugeordneten Speicherbereichs* + 1.
In der Praxis ist es nahezu unmöglich, die obere Grenze selbst zu ermitteln. Es stehen Verwaltungssichten und Tabellenfunktionen zur Verfügung, die zur Ermittlung der aktuellen oberen Grenze verwendet werden können. Jedoch kann sich diese aufgrund von Zeilenoperationen jeden Augenblick ändern.

Beachten Sie, dass die obere Grenze kein Anzeiger für die Anzahl der belegten Seiten ist, da einige Speicherbereiche unterhalb der oberen Grenze möglicherweise durch Löschen von Daten geleert wurden. Auch in diesem Fall gibt die obere Grenze die höchste zugeordnete Seite im Tabellenbereich an, selbst wenn freie Seiten unterhalb dieser Grenze vorhanden sind.

Sie können die obere Grenze eines Tabellenbereichs senken, indem Sie die Speicherbereiche durch eine Operation zur Verkleinerung von Tabellenbereichen konsolidieren.

Beispiel

Abb. 16 auf Seite 194 zeigt eine Reihe zugeordneter Speicherbereiche in einem Tabellenbereich.

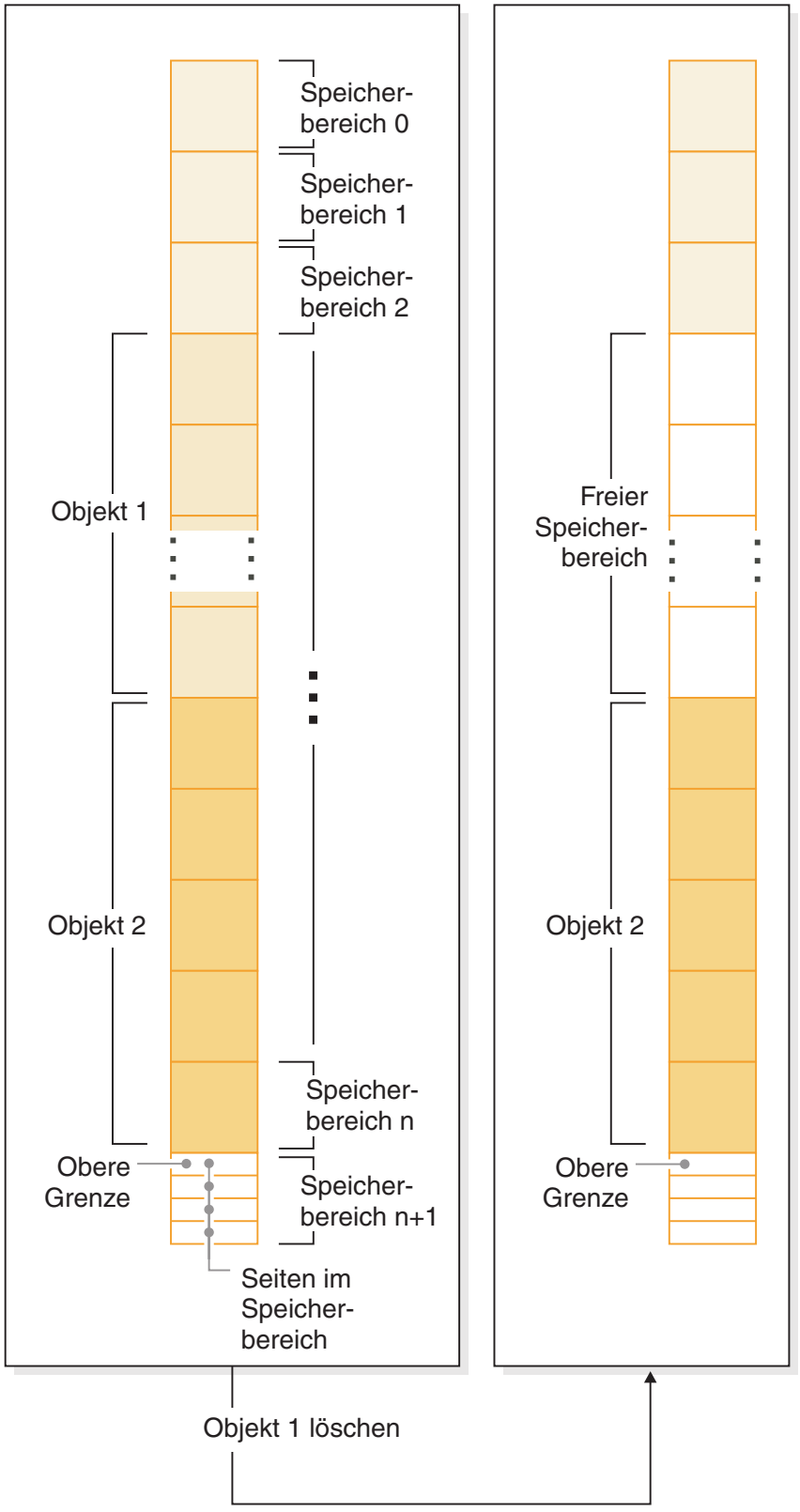


Abbildung 16. Obere Grenze

Wenn ein Objekt gelöscht wird, wird Speicher im Tabellenbereich freigegeben. Allerdings verbleibt die obere Grenze an der vorherigen Stelle, bis eine Art von Speicherkonsolidierungsoperation ausgeführt wird. Je nachdem, wie dem Container neue Speicherbereiche hinzugefügt werden, könnte sie sogar nach oben verschoben werden.

Konsolidierbarer Speicherbereich

Konsolidierbarer Speicherbereich ist eine Funktion für Tabellenbereiche mit dynamischem Speicher für nicht temporäre Tabellen sowie für DMS-Tabellenbereiche in DB2 V9.7. Mit dieser Funktion können Sie verwendeten Speicher unterhalb der *oberen Grenze* konsolidieren und nicht belegte Speicherbereiche in Ihrem Tabellenbereich zur Wiederverwendung durch das System verfügbar machen.

Bei Tabellenbereichen, die vor DB2 V9.7 erstellt wurden, bestand die einzige Möglichkeit, Speicher für das System freizugeben, darin, Container zu löschen oder die Größe von Containern zu verringern, indem nicht verwendete, EXTENTSIZE große Speicherbereiche *oberhalb* der oberen Grenze beseitigt wurden. Es gab keinen direkten Mechanismus zur Senkung der oberen Grenze. Sie konnte gesenkt werden, indem Daten entladen und erneut in einen leeren Tabellenbereich geladen wurden oder indem indirekte Operationen wie die Ausführung von Tabellen- und Indexreorganisationen ausgeführt wurden. Bei dieser letzten Methode war es dennoch möglich, dass die obere Grenze nicht gesenkt werden konnte, auch wenn freie Speicherbereiche unterhalb dieser Grenze vorhanden waren.

Während des Prozesses zur Konsolidierung von Speicherbereichen werden Speicherbereiche, die Daten enthalten, in nicht verwendete Speicherbereiche unterhalb der oberen Grenze versetzt. Wenn nach der Versetzung von Speicherbereichen weiterhin freie Speicherbereiche unterhalb der oberen Grenze vorhanden sind, werden diese als verfügbarer Speicher freigeben. Anschließend wird die obere Grenze auf die Seite im Tabellenbereich verlegt, die unmittelbar auf den letzten verwendeten Speicherbereich folgt. In Tabellenbereichen, in denen konsolidierbarer Speicher verfügbar ist, können Sie mithilfe der Anweisung ALTER TABLESPACE nicht verwendete Speicherbereiche freigeben. Abb. 17 auf Seite 196 bietet eine Übersicht über die Funktionsweise von konsolidierbarem Speicherbereich.

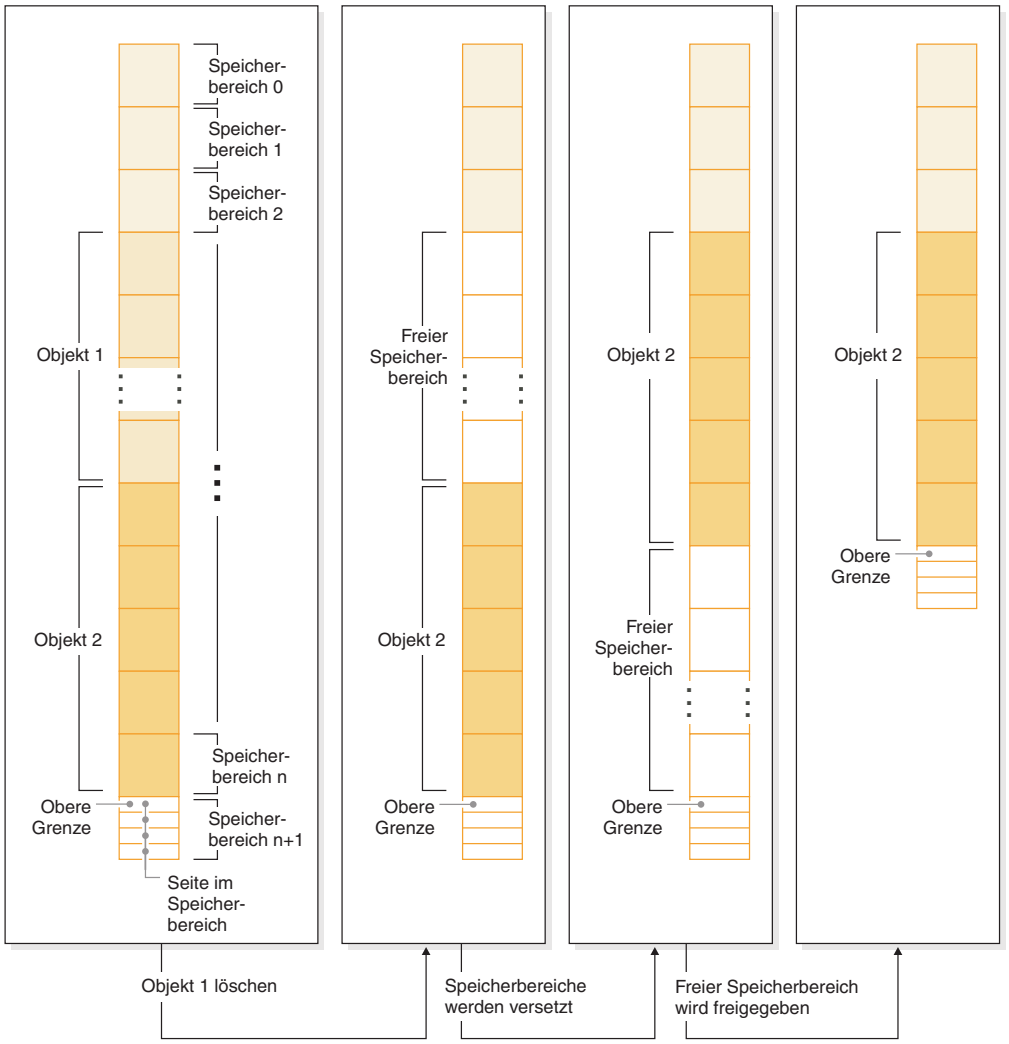


Abbildung 17. Funktionsweise von konsolidierbarem Speicherbereich. Wenn die Funktion des konsolidierbaren Speicherbereichs für einen Tabellenbereich aktiviert ist, können mit Daten gefüllte Speicherbereiche in leere Speicherbereiche weiter unten im Tabellenbereich versetzt werden.

Die Funktion zur Konsolidierung von Speicherbereichen unterhalb der oberen Grenze wird von allen Tabellenbereichen mit dynamischem Speicher für nicht temporäre Tabellen und von DMS-Tabellenbereichen, die in DB2 Version 9.7 und späteren Versionen erstellt wurden, zur Verfügung gestellt. Für Tabellenbereiche, die in einer früheren Version erstellt wurden, müssen Sie den Tabellenbereich zunächst durch einen Tabellenbereich ersetzen, der mit DB2 V9.7 erstellt wurde. Sie können die Daten entweder entladen und erneut laden oder sie durch eine Operation zur Onlineversetzung von Tabellen mithilfe der Prozedur SYSPROC.ADMIN_MOVE_TABLE versetzen. Eine solche Migration ist jedoch nicht erforderlich. Tabellenbereiche, für die die Funktion des konsolidierbaren Speicherbereichs aktiviert ist, können in derselben Datenbank mit Tabellenbereichen koexistieren, die über keinen konsolidierbaren Speicherbereich verfügen.

Die Verkleinerung von Tabellenbereichen durch Versetzen von Speicherbereichen ist eine Onlineoperation. Das heißt, dass DML-Anweisungen (DML, Datenbearbeitungssprache) und DDL-Anweisungen (DDL, Datendefinitionssprache) weiterhin ausgeführt werden können, während die Reduzierungsoperation stattfindet. Einige Operationen, wie zum Beispiel BACKUP oder RESTORE, können nicht gleichzei-

tig mit Operationen zum Versetzen von Speicherbereichen ausgeführt werden. In diesen Fällen wartet der Prozess, der Zugriff auf die Speicherbereiche benötigt (z. B. BACKUP), bis eine Anzahl von Speicherbereichen (die nicht durch den Benutzer konfigurierbar ist) versetzt wurde. Dann erhält der Backup-Prozess eine Sperre für die fraglichen Speicherbereiche und setzt die Verarbeitung an diesem Punkt fort.

Sie können den Fortschritt der Speicherbereichsversetzung mithilfe der Tabellenfunktion MON_GET_EXTENT_MOVEMENT_STATUS überwachen.

Tipp: Zur Maximierung der Größe des Speicherbereichs, der durch die Anweisung ALTER TABLESPACE konsolidiert wird, führen Sie zunächst eine REORG-Operation für die Tabellen und Indizes im betreffenden Tabellenbereich aus.

Tabellenbereiche mit dynamischem Speicher

Sie können die Größe von Tabellenbereichen mit dynamischem Speicher auf verschiedene Arten verkleinern:

Nur Verkleinerung von Containern
Bei dieser Option werden keine Speicherbereiche versetzt. Der Datenbankmanager versucht, die Größe der Container zu verringern, indem er zunächst EXTENTSIZE große Speicherbereiche freigibt, für die Löschoperationen anstehen. (Es ist möglich, dass einige Speicherbereiche mit dem Status „Löschen anstehend" aus Wiederherstellbarkeitsgründen nicht freigegeben werden können, sodass sie verbleiben.) Wenn sich die obere Grenze unter den freigegebenen Speicherbereichen befand, wird sie gesenkt. Andernfalls erfolgt keine Änderung an der oberen Grenze. Als Nächstes wird die Größe der Container angepasst, sodass die Gesamtgröße des Speichers im Tabellenbereich gleich oder unwesentlich größer als die obere Grenze ist. Diese Operation wird durch die Anweisung ALTER TABLESPACE allein mit der Klausel REDUCE ausgeführt.

Nur Senkung der oberen Grenze
Bei dieser Option wird die maximale Anzahl von Speicherbereichen versetzt, um die obere Grenze zu senken, jedoch werden keine Operationen zur Größenänderung von Containern ausgeführt. Diese Operation wird durch die Anweisung ALTER TABLESPACE allein mit der Klausel LOWER HIGH WATER MARK ausgeführt.

Senkung der oberen Grenze und Verkleinerung von Containern um einen bestimmten Betrag
Mit dieser Option können Sie einen absoluten Betrag in KB, MB oder GB angeben, um den der Tabellenbereich zu verkleinern ist. Alternativ können Sie einen relativen Betrag zur Verkleinerung angeben, indem Sie einen Prozentsatz eingeben. In beiden Fällen versucht der Datenbankmanager zunächst, den Speicherplatz um den angeforderten Betrag zu verkleinern, ohne EXTENTSIZE große Speicherbereiche zu versetzen. Das heißt, er versucht, den Tabellenbereich lediglich durch eine Verkleinerung der Containergröße zu verringern (siehe Beschreibung unter Nur Verkleinerung von Containern), indem er Speicherbereiche mit anstehendem Löschen freigibt und versucht, die obere Grenze herabzusetzen. Wenn diese Methode keine ausreichende Verkleinerung erzielt, beginnt der Datenbankmanager anschließend, belegte Speicherbereiche weiter nach unten im Tabellenbereich zu versetzen, um die obere Grenze zu senken. Wenn die Speicherbereichsversetzung abgeschlossen ist, wird die Größe der Container so geändert, dass die Gesamtgröße des Speichers im Tabellenbereich gleich oder unwesentlich größer als die obere Grenze ist. Wenn der Tabellenbereich nicht um

den angeforderten Betrag verkleinert werden kann, weil nicht genügend Speicherbereiche vorhanden sind, die versetzt werden können, wird die obere Grenze so weit wie möglich herabgesetzt. Diese Operation wird mithilfe der Anweisung ALTER TABLESPACE mit der Klausel REDUCE und Angabe eines bestimmten Betrags, um den der Tabellenbereich zu verkleinern ist, ausgeführt.

Senkung der oberen Grenze und Verkleinerung von Containern um den größtmöglichen Betrag
In diesem Fall versetzt der Datenbankmanager so viele Speicherbereiche wie möglich, um die Größe des Tabellenbereichs und seiner Container zu verringern. Diese Operation wird durch die Anweisung ALTER TABLESPACE mit der Klausel REDUCE MAX ausgeführt.

Wenn der Prozess der Speicherbereichsversetzung gestartet wurde, können Sie ihn durch die Anweisung ALTER TABLESPACE mit der Klausel REDUCE STOP stoppen. Alle Speicherbereiche, die versetzt wurden, werden festgeschrieben, die obere Grenze wird auf einen möglichst niedrigen Wert herabgesetzt und die Containergröße wird an den neuen, niedrigeren Wert für die obere Grenze angepasst.

DMS-Tabellenbereiche

DMS-Tabellenbereiche können auf zwei Arten verkleinert werden:

Nur Verkleinerung von Containern
Bei dieser Option werden keine Speicherbereiche versetzt. Der Datenbankmanager versucht, die Größe der Container zu verringern, indem er zunächst EXTENTSIZE große Speicherbereiche freigibt, für die Löschoperationen anstehen. (Es ist möglich, dass einige Speicherbereiche mit dem Status „Löschen anstehend" aus Wiederherstellbarkeitsgründen nicht gelöscht werden können, sodass sie verbleiben.) Wenn sich die obere Grenze unter den freigegebenen Speicherbereichen befand, wird sie gesenkt. Andernfalls erfolgt keine Änderung an der oberen Grenze. Als Nächstes wird die Größe der Container angepasst, sodass die Gesamtgröße des Speichers im Tabellenbereich gleich oder unwesentlich größer als die obere Grenze ist. Diese Operation wird durch die Anweisung ALTER TABLESPACE allein mit der Klausel REDUCE *datenbankcontainer* ausgeführt.

Nur Senkung der oberen Grenze
Bei dieser Option wird die maximale Anzahl von Speicherbereichen versetzt, um die obere Grenze zu senken, jedoch werden keine Operationen zur Größenänderung von Containern ausgeführt. Diese Operation wird durch die Anweisung ALTER TABLESPACE allein mit der Klausel LOWER HIGH WATER MARK ausgeführt.

Das Senken der oberen Grenze und die Verringerung der Containergröße ist bei Tabellenbereichen mit dynamischem Speicher eine kombinierte, automatische Operation. Bei DMS-Tabellenbereichen müssen Sie hingegen zwei Operationen ausführen, um beide Ziele zu erreichen, d. h. eine niedrigere obere Grenze und geringere Containergrößen:

1. Zunächst müssen Sie die obere Grenze für den Tabellenbereich senken, indem Sie die Anweisung ALTER TABLESPACE mit der Klausel LOWER HIGH WATER MARK verwenden.
2. Als Nächstes müssen Sie die Anweisung ALTER TABLESPACE allein mit der Klausel REDUCE *datenbankcontainer* verwenden, um die Operationen zur Anpassung der Containergrößen auszuführen.

Wenn der Prozess der Speicherbereichsversetzung gestartet wurde, können Sie ihn durch die Anweisung ALTER TABLESPACE mit der Klausel LOWER HIGH WATER MARK STOP stoppen. Alle Speicherbereiche, die versetzt wurden, werden festgeschrieben, die obere Grenze wird auf den neuen Wert herabgesetzt.

Beispiele

Beispiel 1: Verringern der Größe eines Tabellenbereichs mit dynamischem Speicher um den maximalen Betrag

Angenommen, in einer Datenbank mit einem Tabellenbereich TS mit dynamischem Speicher und drei Tabellen T1, T2 und T3 werden die Tabellen T1 und T3 gelöscht:

```
DROP TABLE T1
DROP TABLE T3
```

Unter der Annahme, dass die Speicherbereiche jetzt nicht mehr belegt sind, bewirkt nun die folgende Anweisung, dass die zuvor von den Tabellen T1 und T3 belegten Speicherbereiche wieder freigegeben und die obere Grenze des Tabellenbereichs gesenkt wird:

```
ALTER TABLESPACE TS REDUCE MAX
```

Beispiel 2: Verringern der Größe eines Tabellenbereichs mit dynamischem Speicher um einen bestimmten Betrag

Angenommen, eine Datenbank hat einen Tabellenbereich TS mit dynamischem Speicher und enthält zwei Tabellen T1 und T2. Die Tabelle T1 wird nun gelöscht:

```
DROP TABLE T1
```

Um jetzt die Größe des Tabellenbereichs um 1 MB zu verringern, kann die folgende Anweisung verwendet werden:

```
ALTER TABLESPACE TS REDUCE SIZE 1M
```

Alternativ könnte der Tabellenbereich durch eine Anweisung wie die folgende um einen Prozentsatz seiner bestehenden Größe verkleinert werden:

```
ALTER TABLESPACE TS REDUCE SIZE 5 PERCENT
```

Beispiel 3: Verringern der Größe eines Tabellenbereichs mit dynamischem Speicher, wenn freier Speicher unterhalb der oberen Grenze vorhanden ist

Wie in Beispiel 1 wird von einer Datenbank ausgegangen, die einen Tabellenbereich TS mit dynamischem Speicher besitzt und drei Tabellen T1, T2 und T3 enthält. Dieses Mal werden die Tabellen T2 und T3 gelöscht, sodass eine Gruppe von fünf freien Speicherbereichen unmittelbar unterhalb der oberen Grenze vorhanden ist. Unter der Annahme, dass in diesem Fall jeder Speicherbereich aus zwei 4-KB-Seiten besteht, sind nun tatsächlich 40 KB freier Speicher direkt unterhalb der oberen Grenze vorhanden. Jetzt wird eine Anweisung wie die folgende ausgeführt:

```
ALTER TABLESPACE TS REDUCE SIZE 32K
```

Der Datenbankmanager kann daraufhin die obere Grenze senken und die Containergröße verringern, ohne Speicherbereiche versetzen zu müssen. Dieses Szenario wird in Abb. 18 auf Seite 200 veranschaulicht.

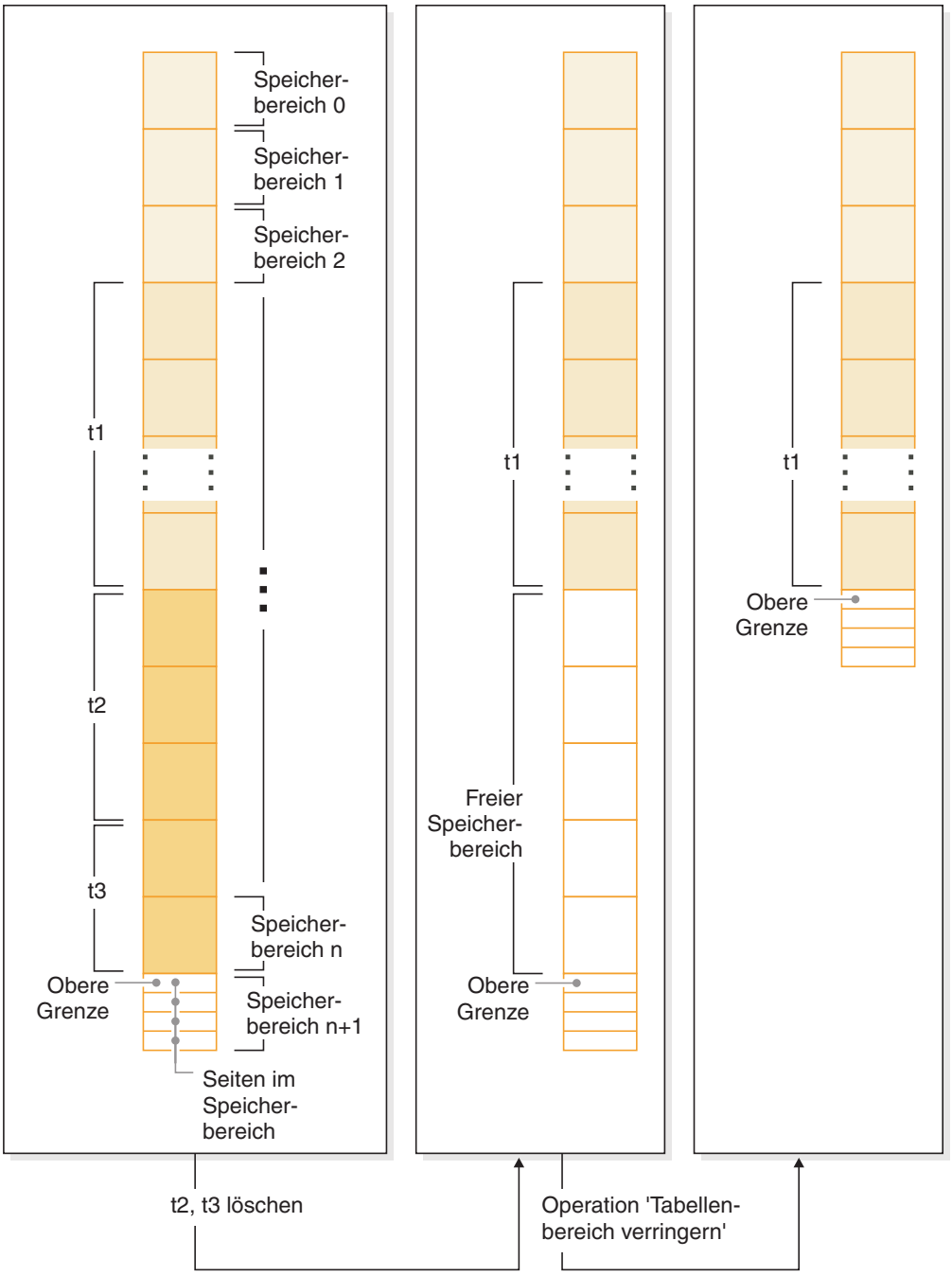


Abbildung 18. Senken der oberen Grenze, ohne Speicherbereiche versetzen zu müssen

Beispiel 4: Verringern der Größe eines DMS-Tabellenbereichs

Angenommen, eine Datenbank hat einen DMS-Tabellenbereich TS und enthält drei Tabellen T1, T2 und T3. Die Tabellen T1 und T3 werden nun gelöscht:

```
DROP TABLE T1
DROP TABLE T3
```

Das Senken der oberen Grenze und das Verringern der Containergröße ist bei einem DMS-Tabellenbereich eine Operation in zwei Schritten. Zuerst senken Sie die obere Grenze durch Versetzen von Speicherbereichen mithilfe der folgenden Anweisung:

```
ALTER TABLESPACE TS LOWER HIGH WATER MARK
```

Anschließend verringern Sie die Größe der Container mit einer Anweisung wie der folgenden:

```
ALTER TABLESPACE TS REDUCE (ALL CONTAINERS 5 M)
```

Vergleich von SMS-Tabellenbereichen, DMS-Tabellenbereichen und Tabellenbereichen mit dynamischem Speicher

SMS-Tabellenbereiche, DMS-Tabellenbereiche und Tabellenbereiche mit dynamischem Speicher bieten verschiedene Funktionen, die in verschiedenen Situationen von Vorteil sein können.

Wichtig: Die Klausel MANAGED BY SYSTEM in der Anweisung CREATE TABLESPACE für Benutzertabellenbereiche wird möglicherweise in einem zukünftigen Release entfernt. Verwenden Sie stattdessen die Klausel MANAGE BY AUTOMATIC STORAGE oder MANAGE BY DATABASE.

Tabelle 12. Vergleich zwischen SMS-Tabellenbereichen, DMS-Tabellenbereichen und Tabellenbereichen mit dynamischem Speicher

	SMS-Tabellenbereiche	**DMS-Tabellenbereiche**	**Tabellenbereiche mit dynamischem Speicher**
Erstellung	Werden mit der Klausel MANAGED BY SYSTEM der Anweisung CREATE TABLESPACE erstellt.	Werden mit der Klausel MANAGED BY DATABASE der Anweisung CREATE TABLESPACE erstellt.	Werden mit der Klausel MANAGED BY AUTOMATIC STORAGE der Anweisung CREATE TABLESPACE bzw. durch Weglassen der gesamten Klausel MANAGED BY erstellt. Wenn der dynamische Speicher bei der Erstellung der Datenbank aktiviert wurde, wird jeder Tabellenbereich, den Sie erstellen, standardmäßig als Tabellenbereich mit dynamischem Speicher erstellt, sofern Sie nicht anderes angeben.
Erstdefinition und -position für Container	Erfordert, dass Container in Form eines Verzeichnisnamens definiert werden.	• Erfordert, dass Container in Form von Dateien oder Einheiten definiert werden. • Die Anfangsgröße für jeden Container muss angegeben werden.	Sie geben beim Erstellen eines Tabellenbereichs mit dynamischem Speicher keine Liste von Containern an. Stattdessen erstellt der Datenbankmanager Container automatisch in allen Speicherpfaden, die der Datenbank zugeordnet wurden. Die Daten werden gleichmäßig auf alle Container verteilt, sodass die Speicherpfade gleichmäßig ausgelastet werden.

Tabelle 12. Vergleich zwischen SMS-Tabellenbereichen, DMS-Tabellenbereichen und Tabellenbereichen mit dynamischem Speicher (Forts.)

	SMS-Tabellenbereiche	**DMS-Tabellenbereiche**	**Tabellenbereiche mit dynamischem Speicher**
Erstzuordnung von Speicher	Erfolgt nach Bedarf. Da das Dateisystem die Zuordnung von Speicher steuert, ist die Wahrscheinlichkeit geringer, dass Seiten zusammenhängend zugeordnet werden. Dies könnte sich auf die Leistung einiger Typen von Abfragen auswirken.	Erfolgt bei der Erstellung des Tabellenbereichs. • Speicherbereiche sind mit höherer Wahrscheinlichkeit zusammenhängend als bei SMS-Tabellenbereichen. • Seiten innerhalb von Speicherbereichen sind bei Einheitencontainern immer zusammenhängend.	• Für Tabellenbereiche mit dynamischem Speicher für nicht temporäre Tabellen gilt: – Der Speicher wird bei der Erstellung des Tabellenbereichs zugeordnet. – Sie können die Anfangsgröße für den Tabellenbereich angeben. • Für Tabellenbereiche mit dynamischem Speicher für temporäre Tabellen wird Speicher nach Bedarf zugeordnet.
Änderungen an Tabellenbereichscontainern	Keine Änderungen nach der Erstellung möglich. Ausnahme: Hinzufügen von Containern für neue Datenpartitionen, wenn diese hinzugefügt werden.	• Container können erweitert oder hinzugefügt werden. Eine Neuverteilung der Daten des Tabellenbereichs erfolgt, wenn der neue Speicher unterhalb der oberen Grenze für den Tabellenbereich hinzugefügt wird. • Container können verkleinert oder gelöscht werden. Eine Neuverteilung erfolgt, wenn Daten in dem Speicher, der gelöscht wird, enthalten sind.	• Container können gelöscht oder verkleinert werden, wenn die Tabellenbereichsgröße verringert wird. • Die Daten des Tabellenbereichs können neu verteilt werden, um sie gleichmäßig auf Container zu verteilen, wenn der Datenbank neuer Speicher hinzugefügt oder Speicher aus der Datenbank gelöscht wird.
Handhabung höheren Speicherbedarfs	Container wachsen, bis sie die Kapazität erreichen, die durch das Dateisystem festgelegt ist. Der Tabellenbereich gilt dann als voll, wenn einer der Container seine maximale Kapazität erreicht.	Container können über die zu Anfang zugeordnete Größe hinaus manuell oder automatisch (falls die automatische Größenänderung aktiviert ist) bis zu den Begrenzungen erweitert werden, die durch das Dateisystem festgelegt sind.	• Container werden automatisch bis zu den Begrenzungen erweitert, die durch das Dateisystem festgelegt sind. • Falls der Datenbank Speicherpfade hinzugefügt werden, werden Container automatisch erweitert bzw. erstellt.
Möglichkeit, verschiedene Typen von Objekten in verschiedenen Tabellenbereichen zu speichern	Nur bei partitionierten Tabellen können sich Indizes und Indexpartitionen in einem Tabellenbereich befinden, der vom Tabellenbereich mit den Tabellendaten separat ist.	Tabellen, Speicher für zugehörige große Objekte (LOBs) und Indizes können sich jeweils in separaten Tabellenbereichen befinden.	Tabellen, Speicher für zugehörige große Objekte (LOBs) und Indizes können sich jeweils in separaten Tabellenbereichen befinden.

Tabelle 12. Vergleich zwischen SMS-Tabellenbereichen, DMS-Tabellenbereichen und Tabellenbereichen mit dynamischem Speicher (Forts.)

	SMS-Tabellenbereiche	**DMS-Tabellenbereiche**	**Tabellenbereiche mit dynamischem Speicher**
Laufender Wartungsbedarf	Keiner	• Hinzufügen oder Erweitern von Containern • Löschen oder Verkleinern von Containern • Senken der oberen Grenze • Neuverteilung	• Verringern der Größe des Tabellenbereichs • Senken der oberen Grenze • Neuverteilung
Verwenden von RESTORE zur Neudefinition von Containern	Sie können eine umgeleitete Restoreoperation verwenden, um die dem Tabellenbereich zugeordneten Container erneut zu definieren.	Sie können eine umgeleitete Restoreoperation verwenden, um die dem Tabellenbereich zugeordneten Container erneut zu definieren.	Sie können keine umgeleitete Restoreoperation dazu verwenden, die dem Tabellenbereich zugeordneten Container erneut zu definieren, da der Datenbankmanager den Speicherbereich verwaltet.
Leistung	Im Allgemeinen langsamer als DMS-Tabellenbereiche und Tabellenbereiche mit dynamischem Speicher, insbesondere für große Tabellen.	SMS-Tabellenbereichen im Allgemeinen überlegen.	Ähnlich wie DMS-Tabellenbereiche.

Von den drei Tabellenbereichstypen sind Tabellenbereiche mit dynamischem Speicher am einfachsten einzurichten und zu pflegen und werden für die meisten Anwendungen empfohlen. Sie sind in folgenden Situationen besonders vorteilhaft:

- Sie haben größere Tabellen oder Tabellen, für die ein rasches Wachstum zu erwarten ist.
- Sie wollen nicht regelmäßig Entscheidungen dazu treffen müssen, wie das Containerwachstum zu verwalten ist.
- Sie wollen in der Lage sein, verschiedene Typen zugehöriger Objekte (z. B. Tabellen, LOB-Daten, Indizes) in verschiedenen Tabellenbereichen zu speichern, um die Leistung zu verbessern.

DMS-Tabellenbereiche sind in folgenden Situationen nützlich:

- Sie haben größere Tabellen oder Tabellen, für die ein rasches Wachstum zu erwarten ist.
- Sie wollen größere Kontrolle darüber haben, wo Daten physisch gespeichert werden.
- Sie wollen in der Lage sein, die Verwendung von Speicher (z. B. das Hinzufügen von Containern) anzupassen oder zu steuern.
- Sie wollen in der Lage sein, verschiedene Typen zugehöriger Objekte (z. B. Tabellen, LOB-Daten, Indizes) in verschiedenen Tabellenbereichen zu speichern, um die Leistung zu verbessern.

SMS-Tabellenbereiche sind in folgenden Situationen nützlich:

- Sie haben kleinere Tabellen, für die kein rasches Wachstum zu erwarten ist.
- Sie wollen größere Kontrolle darüber haben, wo Daten physisch gespeichert werden.
- Sie wollen in Bezug auf die Containerverwaltung keinen großen Aufwand treiben.

- Es ist nicht erforderlich, verschiedene Typen zugehöriger Objekte (z. B. Tabellen, LOB-Daten, Indizes) in verschiedenen Tabellenbereichen zu speichern. (Nur für partitionierte Tabellen *können* Indizes in von den Tabellendaten separaten Tabellenbereichen gespeichert werden.)

Überlegungen zur Auslastung beim Entwerfen von SMS- und DMS-Tabellenbereichen:

Der primäre Typ der Auslastung, die vom Datenbankmanager in Ihrer Umgebung verwaltet wird, kann Ihre Wahl des zu verwendenden Typs von Tabellenbereich und der anzugebenden Seitengröße beeinflussen.

Eine OLTP-Auslastung (Onlinetransaktionsverarbeitung) ist durch Transaktionen charakterisiert, die einen wahlfreien Zugriff auf Daten benötigen, häufige Einfüge- oder Aktualisierungsaktivitäten verursachen und Abfragen enthalten, die in der Regel nur kleine Datenmengen zurückliefern. In Anbetracht dessen, dass der Zugriff wahlfrei nur auf eine bzw. wenige Seiten erfolgt, ist ein Vorablesezugriff weniger wahrscheinlich.

In diesem Fall zeigen DMS-Tabellenbereiche mit Einheitencontainern die beste Leistung. DMS-Tabellenbereiche mit Dateicontainern sind ebenfalls sinnvolle Möglichkeiten für eine OLTP-Auslastung, wenn es nicht auf maximale Leistung ankommt. Beachten Sie, dass die Verwendung von DMS-Tabellenbereichen mit Dateicontainern (wobei FILE SYSTEM CACHING auf OFF gesetzt ist) auf einer Ebene durchgeführt werden kann, die mit unformatierten DMS-Tabellenbereichscontainern vergleichbar ist. Wenn nur wenige oder gar keine sequenziellen E/A-Operationen zu erwarten sind, spielen die Werte der Parameter EXTENTSIZE und PREFETCHSIZE in der Anweisung CREATE TABLESPACE keine wichtige Rolle für die E/A-Effizienz. Wichtig ist jedoch, eine ausreichende Anzahl von Seitenlöschfunktionen über den Konfigurationsparameter *chngpgs_thresh* festzulegen.

Eine Abfrageauslastung ist durch Transaktionen charakterisiert, die einen sequenziellen oder teilweise sequenziellen Zugriff auf Daten benötigen und in der Regel große Datenmengen zurückliefern. Ein DMS-Tabellenbereich mit mehreren Einheitencontainern (wobei sich jeder Container auf einer separaten Platte befindet) bietet das größte Potenzial für einen effizienten parallelen Vorablesezugriff. Der Wert des Parameters PREFETCHSIZE in der Anweisung CREATE TABLESPACE sollte das Produkt aus dem Wert des Parameters EXTENTSIZE multipliziert mit der Anzahl der Einheitencontainer sein. Alternativ dazu können Sie eine Vorablesezugriffsgröße von -1 angeben, und der Datenbankmanager wählt automatisch eine entsprechende Vorablesezugriffsgröße aus. Dadurch kann der Datenbankmanager aus allen Containern parallel vorablesen. Wenn sich die Anzahl von Containern ändert oder der Vorablesezugriff aus einem Grund verstärkt oder verringert werden muss, kann der Wert für PREFETCHSIZE über die Anweisung ALTER TABLESPACE entsprechend geändert werden.

Eine sinnvolle Alternative für eine Abfrageauslastung ist die Verwendung von Dateien, wenn das Dateisystem über eine eigene Vorablesefunktion verfügt. Die Dateien können entweder zu DMS-Tabellenbereichen mit Dateicontainern gehören oder Dateien für SMS-Tabellenbereiche sein. Beachten Sie, dass Sie bei Verwendung von SMS sicherstellen müssen, dass die Verzeichniscontainer getrennten physischen Datenträgern zugeordnet sind, um E/A-Parallelität zu erreichen.

Wichtig: Benutzertabellenbereiche, die vom Betriebssystem verwaltete Speicherbereiche (System Managed Space, SMS) verwenden, sind veraltet und werden in einem zukünftigen Release möglicherweise entfernt. Verwenden Sie stattdessen vom

Datenbankmanager verwaltete Tabellenbereiche (Database Managed Spaces, DMS) oder Tabellenbereiche mit dynamischem Speicher (Automatic Storage Spaces, AMS).

Das Ziel für eine gemischte Auslastung besteht darin, einzelne E/A-Anforderungen so effizient wie möglich für OLTP-Auslastungen zu machen und die Effizienz paralleler E/A-Operationen für Abfrageauslastungen zu maximieren.

Beim Ermitteln der Seitengröße für einen Tabellenbereich sind folgende Überlegungen zu berücksichtigen:

- Für OLTP-Anwendungen, die wahlfreie Lese- und Schreiboperationen durchführen, ist eine geringere Seitengröße empfehlenswert, weil dabei kein Pufferspeicher durch unerwünschte Zeilen verschwendet wird.
- Für DSS-Anwendungen (Decision Support System), die jeweils auf eine große Anzahl aufeinander folgender Zeilen gleichzeitig zugreifen, sind größere Seiten empfehlenswert, weil dadurch weniger E/A-Anforderungen erforderlich sind, um eine bestimmte Anzahl Zeilen zu lesen.
- Durch größere Seiten können Sie möglicherweise die Zahl der Indexstufen reduzieren.
- Größere Seiten unterstützen längere Zeilen.
- Auf 4-KB-Standardseiten sind Tabellen auf 500 Spalten begrenzt, während auf größeren Seiten (8 KB, 16 KB und 32 KB) 1.012 Spalten unterstützt werden.
- Die Maximalgröße des Tabellenbereichs ist proportional zur Seitengröße des Tabellenbereichs.

Hinweise zu SMS- und DMS-Einheiten:

Bei der Entscheidung über die Verwendung von Dateisystemdateien bzw. Einheiten für Tabellenbereichscontainer sind zwei Dinge zu berücksichtigen: das Puffern von Daten und die Verwendung von LOB- bzw. LONG-Daten.

- **Puffern von Daten**

  Vom Plattenspeicher gelesene Tabellendaten sind in der Regel im Pufferpool der Datenbank verfügbar. In einigen Fällen kann es geschehen, dass eine Datenseite aus dem Pufferpool entfernt wird, bevor sie von der Anwendung verwendet wurde. Dies ist insbesondere dann möglich, wenn der Pufferpoolbereich für andere Datenseiten benötigt wird. Bei Tabellenbereichen, die SMS- oder DMS-Dateicontainer verwenden, kann durch das Dateisystemcaching Ein-/Ausgabe vermieden werden, die andernfalls erforderlich wäre.

  Tabellenbereiche, die DMS-Einheitencontainer verwenden, verwenden das Dateisystem oder den Cache des Dateisystems nicht. Daher könnten Sie den Pufferpool der Datenbank vergrößern und den Cache des Dateisystems verkleinern, um den Umstand auszugleichen, dass DMS-Tabellenbereiche, die Einheitencontainer verwenden, keine doppelte Pufferung nutzen.

  Wenn Monitortools auf Systemebene zeigen, dass das Aufkommen an E/A-Operationen für einen DMS-Tabellenbereich im Vergleich zu einem äquivalenten SMS-Tabellenbereich höher liegt, kann sich dieser Unterschied durch die doppelte Pufferung erklären.

  Wichtig: Benutzertabellenbereiche, die vom Betriebssystem verwaltete Speicherbereiche (System Managed Space, SMS) verwenden, sind veraltet und werden in einem zukünftigen Release möglicherweise entfernt. Verwenden Sie stattdessen vom Datenbankmanager verwaltete Tabellenbereiche (Database Managed Spaces, DMS) oder Tabellenbereiche mit dynamischem Speicher (Automatic Storage Spaces, AMS).

- **Verwenden von LOB- oder LONG-Daten**

  Wenn eine Anwendung LOB- oder LONG-Daten abruft, verwendet der Datenbankmanager keine seiner Puffer, um die Daten zwischenzuspeichern. Wenn eine Anwendung eine dieser Seiten benötigt, muss der Datenbankmanager sie vom Plattenspeicher abrufen. Wenn LOB- oder LONG-Daten jedoch in DMS-Dateicontainern gespeichert werden, kann die Pufferung durch den Dateisystemcache erfolgen und dadurch eine bessere Leistung erreicht werden.

  Da Systemkataloge einige LOB-Spalten enthalten, sollten Sie sie in DMS-Dateitabellenbereichen speichern.

Tabellenbereiche des Typs TEMPORARY

Tabellenbereiche für temporäre Tabellen enthalten temporäre Daten, die der Datenbankmanager zur Ausführung von Operationen wie Sortierungen und Joins benötigt, da für solche Operationen zusätzlicher Speicherplatz zur Verarbeitung der Ergebnismengen erforderlich ist.

Eine Datenbank benötigt mindestens einen Tabellenbereich für temporäre *Systemtabellen*, der dieselbe Seitengröße wie der Katalogtabellenbereich besitzt. Standardmäßig wird ein Tabellenbereich für temporäre Systemtabellen mit der Bezeichnung TEMPSPACE1 bei der Datenbankerstellung erstellt. Die Standarddatenbankpartitionsgruppe für diesen Tabellenbereich heißt IBMTEMPGROUP. Die Seitengröße für den Tabellenbereich TEMPSPACE1 entspricht dem Wert, der bei der Erstellung der Datenbank selbst angegeben wurde (Standardwert: 4 KB).

Tabellenbereiche für temporäre Benutzertabellen enthalten temporäre Daten aus Tabellen, die mit einer Anweisung DECLARE GLOBAL TEMPORARY TABLE oder CREATE GLOBAL TEMPORARY TABLE erstellt wurden. Tabellenbereiche für temporäre Benutzertabellen werden nicht standardmäßig bei der Erstellung einer Datenbank erstellt. Sie enthalten außerdem instanziierte Versionen von erstellten temporären Tabellen.

Es wird empfohlen, einen einzelnen temporären Tabellenbereich zu definieren, dessen Seitengröße der Seitengröße entspricht, die in den meisten Benutzertabellenbereichen verwendet wird. Dies sollte für typische Umgebungen und Auslastungen geeignet sein. Es kann jedoch vorteilhaft sein, mit unterschiedlichen Konfigurationen für temporäre Tabellenbereiche und Auslastungen zu experimentieren. Sie sollten die folgenden Punkte beachten:

- Auf temporäre Tabellen wird meist gruppenweise und sequenziell zugegriffen. Das heißt, eine Gruppe von Zeilen wird eingefügt oder eine Gruppe sequenzieller Zeilen wird abgerufen. Daher führt eine größere Seitengröße in der Regel zu einer besseren Leistung, da weniger Anforderungen logischer und physischer Seiten erforderlich sind, um eine bestimmte Datenmenge einzulesen.
- Wenn Sie eine Tabelle unter Verwendung eines Tabellenbereichs für temporäre Tabellen reorganisieren, muss die Seitengröße des Tabellenbereichs für temporäre Tabellen mit der Seitengröße der Tabelle übereinstimmen. Deshalb sollten Sie sicherstellen, dass temporäre Tabellenbereiche vorhanden sind, die für jede Seitengröße definiert sind, die von vorhandenen Tabellen verwendet wird, die Sie vielleicht mithilfe eines temporären Tabellenbereichs reorganisieren.

  Sie können eine Reorganisation auch ohne temporären Tabellenbereich ausführen, indem Sie die Tabelle direkt im selben Tabellenbereich reorganisieren. Diese Art der Reorganisation setzt voraus, dass im Tabellenbereich bzw. in den Tabellenbereichen der Tabelle zusätzlicher Speicherbereich für den Reorganisationsprozess vorhanden ist.

- Wenn Sie SMS-Tabellenbereiche für temporäre Systemtabellen verwenden, empfiehlt sich gegebenenfalls die Verwendung der Registrierdatenbankvariablen DB2_SMS_TRUNC_TMPTABLE_THRESH. Durch Löschen werden die für die temporären Systemtabellen erstellten Dateien auf die Größe 0 abgeschnitten. Die Registrierdatenbankvariable DB2_SMS_TRUNC_TMPTABLE_THRESH kann dazu verwendet werden, einen Zugriff auf Dateisysteme zu umgehen und die Dateien potenziell bei einer Größe ungleich null zu belassen, um den Leistungsaufwand zu vermeiden, der mit wiederholten Erweiterungen und Verkürzungen der Dateien verbunden ist.
- Im Allgemeinen wählt das Optimierungsprogramm, wenn temporäre Tabellenbereiche mit unterschiedlichen Seitengrößen vorhanden sind, den temporären Tabellenbereich aus, dessen Pufferpool die größte Anzahl von Zeilen aufnehmen kann (d. h. in den meisten Fällen der größte Pufferpool). In solchen Fällen empfiehlt es sich, einem der temporären Tabellenbereiche einen großen Pufferpool und den übrigen einen kleineren Pufferpool zuzuordnen. Eine solche Pufferpoolzuordnung hilft bei der Sicherstellung einer effizienten Auslastung des Hauptspeichers. Wenn z. B. Ihr Katalogtabellenbereich 4-KB-Seiten verwendet und die übrigen Tabellenbereiche 8-KB-Seiten, könnte sich folgende Konfiguration für temporäre Tabellenbereiche am besten eignen: ein einzelner temporärer 8-KB-Tabellenbereich mit einem großen Pufferpool und ein einzelner 4-KB-Tabellenbereich mit einem kleinen Pufferpool.
- Es bringt im Allgemeinen keinen Vorteil, mehr als einen temporären Tabellenbereich von jeder Seitengröße zu definieren.

Wie bei regulären und großen Tabellenbereichen mit dynamischem Speicher sind auch temporären Tabellenbereichen mit dynamischem Speicher Speichergruppen zugeordnet. Die Speichergruppenzuordnung für temporäre Tabellenbereiche mit dynamischem Speicher kann jedoch nicht geändert werden. Bei dem Versuch, eine REBALANCE-Operation für einen temporären Tabellenbereich mit dynamischem Speicher auszuführen, wird der Fehler SQL0109N zurückgegeben. Um einem temporären Tabellenbereich eine Speichergruppe zuzuordnen, können Sie den temporären Tabellenbereich löschen und ihn anschließend unter Verwendung einer anderen Speichergruppe erneut erstellen. Wenn Sie Speicherpfade zu einer Speichergruppe hinzufügen, können temporäre Tabellenbereiche die neuen Pfade erst ab der nächsten Datenbankaktivierung nutzen.

Überlegungen zur Auswahl von Tabellenbereichen für Tabellen

Beim Ermitteln der Vorgehensweise für die Zuordnung von Tabellen zu Tabellenbereichen sollten Sie die Verteilung Ihrer Tabellen, die Menge der Daten und den Typ der Daten in der Tabelle sowie verwaltungstechnische Fragen berücksichtigen.

Verteilung der Tabellen

Sie müssen zumindest sicherstellen, dass sich der Tabellenbereich, den Sie auswählen, in einer Datenbankpartitionsgruppe mit der gewünschten Verteilung befindet.

Menge der Daten in der Tabelle

Wenn Sie planen, viele kleine Tabellen in einem Tabellenbereich zu speichern, sollten Sie dazu die Verwendung eines SMS-Tabellenbereichs in Betracht ziehen. Die Vorteile von DMS-Tabellenbereichen im Hinblick auf die Effizienz bei E/A-Operationen und Speicherbereichsverwaltung sind bei kleinen Tabellen nicht so bedeutsam. Die Vorteile von SMS-Tabellenbereichen (bei entsprechendem Bedarf) sind bei kleinen Tabellen attraktiver. Wenn eine Ihrer Tabellen größer ist oder Sie einen schnelleren Zugriff auf

die Daten in den Tabellen benötigen, sollte ein DMS-Tabellenbereich mit einem kleinen Wert für EXTENTSIZE in Betracht gezogen werden.

Eventuell empfiehlt es sich, einen separaten Tabellenbereich für jede umfangreiche Tabelle zu verwenden und kleine Tabellen gemeinsam in einem einzigen Tabellenbereich anzulegen. Diese Trennung ermöglicht es Ihnen außerdem, entsprechend der Auslastung des Tabellenbereichs einen geeigneten Wert für den Parameter EXTENTSIZE auszuwählen.

Wichtig: Benutzertabellenbereiche, die vom Betriebssystem verwaltete Speicherbereiche (System Managed Space, SMS) verwenden, sind veraltet und werden in einem zukünftigen Release möglicherweise entfernt. Verwenden Sie stattdessen vom Datenbankmanager verwaltete Tabellenbereiche (Database Managed Spaces, DMS) oder Tabellenbereiche mit dynamischem Speicher (Automatic Storage Spaces, AMS).

Typ der Daten in der Tabelle

Sie könnten beispielsweise über Tabellen mit Langzeitdaten verfügen, die relativ selten verwendet werden und bei denen der Endbenutzer eine längere Antwortzeit für Abfragen, die diese Daten betreffen, vielleicht akzeptiert. In diesem Fall könnten Sie für diese historischen Tabellen einen anderen Tabellenbereich verwenden und diesem Tabellenbereich kostensparendere physische Einheiten mit einer langsameren Zugriffsgeschwindigkeit zuordnen.

Auf der anderen Seite können Sie einige wichtige Tabellen identifizieren, für die eine schnelle Verfügbarkeit und schnelle Antwortzeiten erforderlich sind. Diese Tabellen könnten Sie in einen Tabellenbereich stellen, der einer schnellen physischen Einheit zugeordnet ist, die die Anforderungen für diese wichtigen Daten besser erfüllt.

Wenn Sie mit DMS-Tabellenbereichen arbeiten, können Sie Ihre Tabellendaten auf vier verschiedene Tabellenbereiche verteilen: einen für Indexdaten, einen für Daten großer Objekte (LOB) und Langfelddaten (LF), einen für reguläre Tabellendaten sowie einen für XML-Daten. Auf diese Weise können Sie die Tabellenbereichsmerkmale und die physischen Einheiten der Tabellenbereiche auswählen, die für die Daten am besten geeignet sind. Sie könnten z. B. die Indexdaten auf die schnellste verfügbare Einheit stellen und dadurch eine erhebliche Verbesserung der Leistung erzielen. Wenn Sie eine Tabelle auf DMS-Tabellenbereiche aufteilen, sollten Sie in Betracht ziehen, diese Tabellenbereiche zusammen zu sichern und wiederherzustellen, wenn die aktualisierende Recovery (Rollforward) aktiviert ist. SMS-Tabellenbereiche unterstützen diese Art der Datenverteilung auf Tabellenbereiche nicht.

Verwaltungstechnische Fragen

Einige Verwaltungsfunktionen können auf Tabellenbereichsebene anstatt auf Datenbank- oder Tabellenebene ausgeführt werden. Wenn Sie beispielsweise eine Backupkopie eines Tabellenbereichs anstelle der Backupkopie einer Datenbank erstellen, können Sie Ihre Zeit und Ressourcen besser ausnutzen. Diese Vorgehensweise ermöglicht Ihnen, Tabellenbereiche mit umfangreichen Änderungen häufig zu sichern und von den Tabellenbereichen, die wenig geändert werden, nur gelegentlich neue Backupkopien anzulegen.

Sie können eine Datenbank oder einen Tabellenbereich wiederherstellen. Wenn Tabellen, die sich nicht aufeinander beziehen, keine gemeinsamen

Tabellenbereiche benutzen, haben Sie die Option, kleinere Teile Ihrer Datenbank wiederherzustellen und den Aufwand verringern.

Ein gutes Verfahren besteht darin, Tabellen, die voneinander abhängig sind, in einer Gruppe von Tabellenbereichen zusammenzufassen. Diese Tabellen könnten über referenzielle Integritätsbedingungen oder andere definierte Geschäftsregeln voneinander abhängig sein.

Falls Sie eine bestimmte Tabelle häufig löschen und erneut definieren müssen, können Sie die Tabelle in einem eigenen Tabellenbereich definieren, weil das Löschen eines DMS-Tabellenbereichs effizienter ist als das Löschen einer Tabelle.

Tabellenbereiche ohne Dateisystemcaching

Die empfohlene Methode zur Aktivierung bzw. Inaktivierung der ungepufferten Ein-/Ausgabe unter UNIX, Linux und Windows wird auf der Tabellenbereichsebene ausgeführt.

Dadurch können Sie die ungepufferte Ein-/Ausgabe für bestimmte Tabellenbereiche aktivieren bzw. inaktivieren und gleichzeitig jede Abhängigkeit vom physischen Layout der Datenbank umgehen. Darüber hinaus kann der Datenbankmanager in diesem Fall bestimmen, welche Ein-/Ausgabemethode (gepuffert oder ungepuffert) sich für jede einzelne Datei am besten eignet.

Die Klausel NO FILE SYSTEM CACHING dient zur Aktivierung der ungepufferten Ein-/Ausgabe, sodass gleichzeitig das Dateicaching für einen bestimmten Tabellenbereich inaktiviert wird. Wenn sie aktiviert ist, bestimmt der Datenbankmanager abhängig von der Plattform, welcher Typ von Ein-/Ausgabe, d. h. Direct I/O (DIO) oder Concurrent I/O (CIO), zu verwenden ist. Angesichts der Leistungsverbesserung durch CIO verwendet der Datenbankmanager die CIO-Funktionalität immer dann, wenn sie unterstützt wird. Es gibt keine Benutzerschnittstelle, über die angegeben werden könnte, welche von beiden zu verwenden ist.

Zur Nutzung des maximalen Vorteils der ungepufferten Ein-/Ausgabe kann es notwendig werden, Pufferpools zu vergrößern. Wenn jedoch der Speichermanager mit automatischer Leistungsoptimierung aktiviert und die Pufferpoolgröße auf AUTOMATIC gesetzt wird, führt der Datenbankmanager eine automatische Optimierung der Pufferpoolgröße auf bestmögliche Leistung durch. Beachten Sie, dass diese Funktion erst ab Version 9 verfügbar ist.

Zum Inaktivieren bzw. Aktivieren des Dateisystemcachings geben Sie die Klausel NO FILE SYSTEM CACHING bzw. die Klausel FILE SYSTEM CACHING in der Anweisung CREATE TABLESPACE oder ALTER TABLESPACE an. Bei keiner Angabe einer der beiden Klauseln wird die Standardeinstellung verwendet. Wenn die Anweisung ALTER TABLESPACE ausgeführt wird, müssen vorhandene Verbindungen zur Datenbank erst beendet werden, bevor die neue Caching-Einstellung in Kraft gesetzt wird.

Anmerkung: Wenn ein Attribut abweichend vom Standardwert auf FILE SYSTEM CACHING oder NO FILE SYSTEM CACHING gesetzt wird, ist kein Mechanismus verfügbar, der ein Rücksetzen auf den Standardwert ermöglicht.

Diese Methode der Aktivierung und Inaktivierung des Dateisystemcachings ermöglicht eine Steuerung des Ein-/Ausgabemodus (gepuffert oder ungepuffert) auf der Tabellenbereichsebene.

Zur Feststellung, ob das Dateisystemcaching aktiviert ist, fragen Sie den Wert des Monitorelements **fs_caching** für den Tabellenbereich in der Tabelle MON_GET_TABLESPACE ab.

Alternative Methoden zur Aktivierung/Inaktivierung der ungepufferten Ein-/Ausgabe unter UNIX, Linux und Windows

Einige UNIX-Plattformen unterstützen die Inaktivierung des Caching auf Dateisystemebene durch die Option MOUNT. Weitere Informationen zu dieser Option finden Sie in der Dokumentation zu Ihrem Betriebssystem. Es ist jedoch wichtig, sich über den Unterschied zwischen der Inaktivierung des Cachings auf Tabellenbereichsebene und der Inaktivierung des Cachings auf Dateisystemebene im Klaren zu sein. Auf der Tabellenbereichsebene steuert der Datenbankmanager, welche Dateien mit oder ohne Dateisystemcaching zu öffnen sind. Auf der Dateisystemebene wird jede Datei, die sich in diesem speziellen Dateisystem befindet, ohne Dateisystemcaching geöffnet. Für einige Plattformen, wie zum Beispiel AIX müssen bestimmte Voraussetzungen, wie zum Beispiel die Serialisierung des Lese- und Schreibzugriffs, erfüllt sein, bevor diese Funktion verwendet werden kann. Der Datenbankmanager erfüllt zwar diese Voraussetzungen, jedoch sollten Sie vor der Aktivierung dieser Funktion in der Dokumentation zu Ihrem Betriebssystem alle Informationen zu Voraussetzungen nachlesen, wenn das Zieldateisystem Dateien enthält, die nicht zum Datenbankmanager gehören.

Anmerkung: Die inzwischen veraltete Registrierdatenbankvariable **DB2_DIRECT_IO**, die mit Version 8.1 Fixpack 4 eingeführt wurde, aktiviert kein Dateisystemcaching für alle SMS-Container mit Ausnahme der Tabellenbereiche für Langfelddaten, LOB-Daten und temporäre Tabellendaten unter AIX JFS2. Die Definition dieser Registrierdatenbankvariablen in Version 9.1 oder späteren Versionen hat dieselbe Wirkung wie eine Änderung aller Tabellenbereiche (SMS und DMS) mit der Klausel NO FILE SYSTEM CACHING. Die Verwendung der Variablen **DB2_DIRECT_IO** wird jedoch nicht empfohlen. Es ist beabsichtigt, diese Variable in einem späteren Release zu entfernen. Geben Sie stattdessen NO FILE SYSTEM CACHING auf der Tabellenbereichsebene an.

Alternative Methoden zur Aktivierung/Inaktivierung der ungepufferten Ein-/Ausgabe unter Windows

In früheren Releases konnte die leistungsbezogene Registrierdatenbankvariable **DB2NTNOCACHE** verwendet werden, um das Dateisystemcaching für alle DB2-Dateien zu inaktivieren und dadurch mehr Speicher für die Datenbank zur Verfügung zu stellen, sodass der Pufferpool oder der Sortierspeicher vergrößert werden konnten. Der Unterschied zwischen der Registrierdatenbankvariablen **DB2NTNOCACHE** und der Klausel NO FILE SYSTEM CACHING ist die Möglichkeit, das Caching für Tabellenbereiche selektiv zu inaktivieren. Mit Version 9.5 braucht diese Registrierdatenbankvariable zur Inaktivierung des Dateisystemcachings in der gesamten Instanz nicht mehr definiert zu werden, sofern die Instanz nur neu erstellte Tabellenbereiche enthält, da NO FILE SYSTEM CACHING die Standardeinstellung ist, wenn nicht FILE SYSTEM CACHING explizit angegeben wird.

Leistungsaspekte

Die ungepufferte Ein-/Ausgabe wird im Wesentlichen zur Verbesserung der Leistung verwendet. In einigen Fällen kann jedoch möglicherweise zu Leistungseinbußen kommen, die auf die Kombination einer kleinen Pufferpoolgröße mit einem kleinen Dateisystemcache oder auch auf andere Grün-

de zurückzuführen sind. Zur Leistungsverbesserung können zum Beispiel folgende Maßnahmen empfohlen werden:

- Wenn der Speichermanager mit automatischer Leistungsoptimierung nicht aktiviert ist, aktivieren Sie ihn und setzen die Pufferpoolgröße auf AUTOMATIC, in dem Sie die Anweisung `ALTER BUFFERPOOL name SIZE AUTOMATIC` ausführen. Dies ermöglicht dem Datenbankmanager, die Pufferpoolgröße zu optimieren.
- Wenn der Speichermanager mit automatischer Leistungsoptimierung nicht aktiviert werden soll, vergrößern Sie den Pufferpool in Inkrementen von 10 oder 20 %, bis eine Leistungsverbesserung eintritt.
- Wenn der Speichermanager mit automatischer Leistungsoptimierung nicht aktiviert werden soll, ändern Sie den Tabellenbereich, sodass er „FILE SYSTEM CACHING" verwendet. Dies bewirkt im Wesentlichen, dass die ungepufferte Ein-/Ausgabe inaktiviert wird und der Containerzugriff wieder mit gepufferter Ein-/Ausgabe erfolgt.

Die Leistungsverbesserung sollte vor der Implementierung auf dem Produktionssystem unter kontrollierten Umgebungsbedingungen getestet werden.

Wenn Sie beabsichtigen, Dateisystemdateien anstelle von Einheiten als Tabellenbereichscontainer zu verwenden, sollten Sie das Dateisystemcaching in Betracht ziehen, das wie folgt ausgeführt wird:

- Seiten aus DMS-Dateicontainern (und aus allen SMS-Containern) können vom Betriebssystem im Dateisystemcache zwischengespeichert werden, es sei denn, der Tabellenbereich ist mit NO FILESYSTEM CACHING definiert.
- Seiten aus den Tabellenbereichen von DMS-Einheitencontainern werden vom Betriebssystem nicht im Dateisystemcache zwischengespeichert.

Neue Tabellenbereichscontainer verwenden standardmäßig gleichzeitige Ein-/Ausgabe (CIO) oder direkte Ein-/Ausgabe (DIO)

Der Standard-E/A-Mechanismus für neu erstellte Tabellenbereichscontainer auf den meisten AIX-, Linux-, Solaris- und Windows-Betriebssystemen ist CIO/DIO (Concurrent I/O oder Direct I/O). Diese Standardeinstellung bietet eine Durchsatzsteigerung gegenüber gepufferter Ein-/Ausgabe bei Auslastungen mit intensiver Transaktionsverarbeitung sowie bei Rollbackoperationen.

Das Attribut FILE SYSTEM CACHING bzw. NO FILE SYSTEM CACHING gibt an, ob E/A-Operationen auf der Dateisystemebene im Cache zwischengespeichert werden sollen:

- FILE SYSTEM CACHING gibt an, dass alle E/A-Operationen im Zieltabellenbereich auf der Dateisystemebene im Cache zwischengespeichert werden sollen.
- NO FILE SYSTEM CACHING gibt an, dass alle E/A-Operationen den Cache auf Dateisystemebene umgehen sollen.

Wenn Daten großer Objekte (LOB-Daten) inline gespeichert werden, wird darauf als reguläre Daten zugegriffen. Dafür wird die E/A-Methode (gepuffert oder nicht gepuffert) verwendet, die für das Tabellenbereichsattribut FILE SYSTEM CACHING angegeben wurde.

Wenn LOB-Daten nicht inline gespeichert werden, gelten die folgenden Aussagen:

- Für SMS-Tabellenbereiche wird nicht gepufferter E/A-Zugriff für Langfelddaten (LF-Daten) und für LOB-Daten nicht angefordert, auch wenn das Tabellenbereichsattribut NO FILE SYSTEM CACHING gesetzt ist. Die Pufferung wird im

Dateisystemcache ausgeführt, hängt von Konfiguration und Verhalten des Betriebssystems ab und verbessert möglicherweise die Leistung.

- Bei DMS-Tabellenbereichen unterscheidet DB2 bei der Ausführung von E/A-Operationen nicht zwischen verschiedenen Datentypen. Es erfolgt keine Pufferung von LF- oder LOB-Daten, sofern der Tabellenbereich nicht mit aktiviertem Dateisystemcaching (FILE SYSTEM CACHING) konfiguriert wird. Wenn die Pufferung von LF- oder LOB-Daten in DMS-Tabellenbereichen aus Leistungsgründen gewünscht ist, können Sie diese Daten in einem separaten DMS-Tabellenbereich speichern und das Attribut FILE SYSTEM CACHING explizit aktivieren.

Die folgenden Schnittstellen enthalten das Attribut FILE SYSTEM CACHING:

- Anweisung CREATE TABLESPACE
- Befehl **CREATE DATABASE**
- API sqlecrea() (Feld *sqlfscaching* der Struktur SQLETSDESC)

Wenn dieses Attribut in der Anweisung CREATE TABLESPACE oder im Befehl **CREATE DATABASE** nicht angegeben wird, verarbeitet der Datenbankmanager die Anforderung entsprechend dem Standardverhalten für die Plattform und den Dateisystemtyp. Das genaue Verhalten wird in „Dateisystemcaching - Konfigurationen" beschrieben. Bei der API sqlecrea() weist der Wert `0x2` für das Feld *sqlfscaching* den Datenbankmanager an, die Standardeinstellung zu verwenden.

Beachten Sie, dass der Wert für das Attribut FILE SYSTEM CACHING gegenwärtig von den folgenden Tools interpretiert wird:

- Befehl **GET SNAPSHOT FOR TABLESPACES**
- Befehl **db2pd –tablespaces**
- Befehl `db2look –d dbname –l`

Für den Befehl **db2look** gilt, dass bei keiner Angabe des Attributs FILE SYSTEM CACHING die Ausgabe dieses Attribut nicht enthält.

Beispiel

Nehmen Sie an, dass sich die Datenbank und alle zugehörigen Tabellenbereichscontainer in einem JFS-Dateisystem unter AIX befinden und die folgende Anweisung abgesetzt wird:

```
DB2 CREATE TABLESPACE JFS2
```

Wenn das Attribut nicht angegeben wurde, verwendet der Datenbankmanager das Attribut NO FILE SYSTEM CACHING.

Dateisystemcaching - Konfigurationen

Das Betriebssystem arbeitet standardmäßig mit einem Caching für Dateidaten, die von der Platte gelesen bzw. auf Platte geschrieben werden.

Eine typische Leseoperation führt einen Zugriff auf die physische Platte aus, um Daten von der Platte in den Dateisystemcache einzulesen und die Daten anschließend in den Anwendungspuffer zu kopieren. Analog führt eine Schreiboperation einen Zugriff auf die physische Platte aus, um die Daten aus dem Anwendungspuffer in den Dateisystemcache und anschließend aus dem Cache auf die physische Platte zu kopieren. Auf dieses Verhalten beim Caching von Daten auf der Dateisystemebene bezieht sich die Klausel FILE SYSTEM CACHING der Anweisung CREATE TABLESPACE. Da der Datenbankmanager ein eigenes Datencaching mithilfe von Pufferpools verwaltet, wird das Caching auf Dateisystemebene nicht benötigt, wenn die Größe des Pufferpools entsprechend optimiert ist.

Anmerkung: Der Datenbankmanager verhindert bereits das Caching der meisten DB2-Daten, mit Ausnahme von temporären Daten und LOBs unter AIX; dies geschieht durch die Inaktivierung der Cacheseiten.

In einigen Fällen kann das Caching auf Dateisystemebene und in den Pufferpools zu Leistungseinbußen führen, da für das doppelte Caching zusätzliche CPU-Zyklen benötigt werden. Zur Vermeidung dieses doppelten Cachings besitzen die meisten Dateisysteme eine Funktion, die das Caching auf der Dateisystemebene inaktiviert. Dies wird häufig als *ungepufferte Ein-/Ausgabe* bezeichnet. Unter UNIX wird diese Funktion gemeinhin als *Direct I/O (oder DIO)* bezeichnet. Unter Windows entspricht dies dem Öffnen einer Datei mit der Markierung (Flag) FILE_FLAG_NO_BUFFERING. Darüber hinaus unterstützen einige Dateisysteme wie zum Beispiel IBM JFS2 oder Symantec VERITAS VxFS auch ein erweitertes Direct I/O, nämlich die noch leistungsfähigere Funktion *Concurrent I/O (CIO)*. Der Datenbankmanager unterstützt diese Funktion mit der Tabellenbereichsklausel NO FILE SYSTEM CACHING. Wenn diese definiert ist, nutzt der Datenbankmanager CIO automatisch in Dateisystemen, in denen diese Funktion vorhanden ist. Diese Funktion kann dazu beitragen, den Speicherbedarf des Dateisystemcache zu senken und so mehr Speicher für andere Verwendungszwecke verfügbar zu machen.

Vor Version 9.5 wurde das Schlüsselwort FILE SYSTEM CACHING impliziert, wenn weder NO FILE SYSTEM CACHING noch FILE SYSTEM CACHING angegeben war. Für Version 9.5 gilt, dass bei keiner Angabe eines der Schlüsselwörter standardmäßig NO FILE SYSTEM CACHING verwendet wird. Diese Änderung betrifft nur neu erstellte Tabellenbereiche. Vor Version 9.5 erstellte vorhandene Tabellenbereiche sind nicht betroffen. Diese Änderung gilt für AIX, Linux, Solaris und Windows mit den folgenden Ausnahmen, wobei das Standardverhalten FILE SYSTEM CACHING bleibt:

- AIX JFS
- Solaris (nicht VxFS)
- Linux für System z
- Alle temporären SMS-Tabellenbereichsdateien
- LF-Datendateien (LF = Long Field) und LOB-Datendateien (LOB = Large Object) in permanenten SMS-Tabellenbereichsdateien

Wenn Sie die Standardeinstellung überschreiben möchten, müssen Sie FILE SYSTEM CACHING oder NO FILE SYSTEM CACHING angeben.

Wichtig: Der SMS-Tabellenbereichstyp gilt in Version 10.1 für benutzerdefinierte Tabellenbereiche für persistente Daten als veraltet und wird möglicherweise in einem zukünftigen Release entfernt. Der SMS-Tabellenbereichstyp für Katalogtabellenbereiche und temporäre Tabellenbereiche gilt nicht als veraltet. Weitere Informationen hierzu finden Sie im Abschnitt „SMS-Tabellenbereiche für persistente Daten gelten als veraltet" in *Neuerungen in DB2 Version 10.1*.

Unterstützte Konfigurationen

Tabelle 13 auf Seite 214 zeigt die unterstützte Konfiguration zur Verwendung von Tabellenbereichen ohne Dateisystem-Caching. Außerdem wird noch Folgendes angegeben: (a) Ob die DIO-Funktionalität oder die erweiterte DIO-Funktionalität in allen Fällen verwendet wird; (b) es wird das Standardverhalten angegeben, wenn weder NO FILE SYSTEM CACHING noch FILE SYSTEM CACHING für einen Tabellenbereich auf der Basis der Plattform und des Dateisystemtyps angegeben ist.

Tabelle 13. Unterstützte Konfigurationen für Tabellenbereiche ohne Dateisystemcaching

Plattformen	Dateisystemtyp und erforderliche Mindeststufe	Vom Datenbankmanager übergebene Anforderungen DIO oder CIO, wenn NO FILE SYSTEM CACHING angegeben ist	Standardverhalten, wenn weder NO FILE SYSTEM CACHING noch FILE SYSTEM CACHING angegeben ist
AIX 6.1 und höher	Journal File System (JFS)	DIO	FILE SYSTEM CACHING (Siehe Hinweis 1.)
AIX 6.1 und höher	General Parallel File System (GPFS)	DIO	NO FILE SYSTEM CACHING
AIX 6.1 und höher	Concurrent Journal File System (JFS2)	CIO	NO FILE SYSTEM CACHING
AIX 6.1 und höher	VERITAS Storage Foundation für DB2 4.1 (VxFS)	CIO	NO FILE SYSTEM CACHING
HP-UX Version 11i v3 (Itanium)	VERITAS Storage Foundation 4.1 (VxFS)	CIO	FILE SYSTEM CACHING
Solaris 10, 11	UNIX File System (UFS)	CIO	FILE SYSTEM CACHING (siehe Hinweis 2.)
Solaris 10, 11	VERITAS Storage Foundation für DB2 4.1 (VxFS)	CIO	NO FILE SYSTEM CACHING
Linux-Varianten SLES 10 SP3 oder höher und RHEL 5.2 oder höher (auf folgenden Architekturen: x86, x64, POWER)	ext2, ext3, reiserfs	DIO	NO FILE SYSTEM CACHING
Linux-Varianten SLES 10 SP3 oder höher und RHEL 5.2 oder höher (auf folgenden Architekturen: x86, x64, POWER)	VERITAS Storage Foundation 4.1 (VxFS)	CIO	NO FILE SYSTEM CACHING
Linux-Varianten SLES 10 SP3 oder höher und RHEL 5.2 oder höher (auf folgender Architektur: zSeries)	ext2, ext3 oder reiserfs auf Small Computer System Interface-Platten (SCSI-Platten) mit Fibre Channel Protocol (FCP)	DIO	FILE SYSTEM CACHING
Windows	Keine bestimmte Anforderung; funktioniert mit allen von DB2 unterstützten Dateisystemen.	DIO	NO FILE SYSTEM CACHING

Anmerkung:

1. Unter AIX JFS ist FILE SYSTEM CACHING das Standardverhalten.
2. Unter Solaris UFS ist NO FILE SYSTEM CACHING das Standardverhalten.
3. VERITAS Storage Foundation für den Datenbankmanager hat möglicherweise andere Betriebssystemvoraussetzungen. Bei den oben aufgeführten Plattformen handelt es sich um für das aktuelle Release unterstützte Plattformen. Informationen zu den Voraussetzungen finden Sie bei der Unterstützung zu VERITAS Storage Foundation für DB2.

4. Bei der Verwendung von SFDB2 5.0 anstelle der oben angeführten Mindeststufen muss das Release MP1 RP1 von SFDB2 5.0 verwendet werden. Dieses Release umfasst 5.0-spezifische Fixes.
5. Wenn der Datenbankmanager als Standardeinstellung nicht NO FILE SYSTEM CACHING auswählen soll, müssen Sie FILE SYSTEM CACHING in der entsprechenden SQL, in den entsprechenden Befehlen oder in den entsprechenden APIs angeben.

Beispiele

Beispiel 1: Standardmäßig wird dieser neue Tabellenbereich mit nicht gepufferter Ein-/Ausgabe erstellt. Die Klausel NO FILE SYSTEM CACHING wird impliziert:

```
CREATE TABLESPACE tabellenbereichsname...
```

Beispiel 2: In der folgenden Anweisung gibt die Klausel NO FILE SYSTEM CACHING an, dass die Cachingfunktion auf Dateisystemebene für diesen speziellen Tabellenbereich inaktiviert wird (OFF):

```
CREATE TABLESPACE tabellenbereichsname ... NO FILE SYSTEM CACHING
```

Beispiel 3: Die folgende Anweisung inaktiviert das Caching auf Dateisystemebene für einen vorhandenen Tabellenbereich:

```
ALTER TABLESPACE tabellenbereichsname ... NO FILE SYSTEM CACHING
```

Beispiel 4: Die folgende Anweisung aktiviert das Caching auf Dateisystemebene für einen vorhandenen Tabellenbereich:

```
ALTER TABLESPACE tabellenbereichsname ... FILE SYSTEM CACHING
```

Speicherbereichsgrößen in Tabellenbereichen

Ein *Speicherbereich* (Extent) ist ein Speicherblock in einem Tabellenbereichscontainer. Er stellt die Anzahl von Datenseiten dar, die in einen Container geschrieben werden, bevor in den nächsten Container geschrieben wird. Wenn Sie einen Tabellenbereich erstellen, können Sie die Speicherbereichsgröße (EXTENTSIZE) auf der Basis Ihrer Anforderungen an die Leistung und das Speichermanagement auswählen.

Beim Auswählen einer Speicherbereichsgröße (EXTENTSIZE) ist Folgendes zu beachten:

- Größe und Typ der Tabellen im Tabellenbereich

  In DMS-Tabellenbereichen wird einer Tabelle gleichzeitig jeweils ein EXTENTSIZE großer Speicherbereich zugeordnet. Wenn beim Füllen der Tabelle mit Daten ein EXTENTSIZE großer Bereich voll ist, wird ein neuer Bereich dieser Größe zugeordnet. Speicher in DMS-Tabellenbereichscontainern wird vorab reserviert. Das bedeutet, dass neue EXTENTSIZE große Speicherbereiche zugeordnet werden, bis der Container vollständig gefüllt ist.

  Speicherbereich in SMS-Tabellenbereichen wird einer Tabelle jeweils in der Größe eines EXTENTSIZE-Bereichs oder einer Seite zugeordnet. Wenn beim Füllen der Tabelle mit Daten ein EXTENTSIZE großer Bereich oder eine Seite voll ist, wird ein neuer Bereich oder eine neue Seite zugeordnet, bis alle EXTENTSIZE-Bereiche oder Seiten in dem betreffenden Dateisystem belegt sind. Bei der Verwendung von SMS-Tabellenbereichen ist die mehrseitige Dateizuordnung zulässig. Die mehrseitige Dateizuordnung ermöglicht die Zuordnung jeweils EXTENTSIZE großer Speicherbereiche anstelle einzelner Seiten.

  Die Zuordnung aus mehreren Seiten bestehender Dateien ist standardmäßig aktiviert. Der Wert des Datenbankkonfigurationsparameters **multipage_alloc** gibt an, ob die Zuordnung aus mehreren Seiten bestehender Dateien aktiviert ist.

Anmerkung: Die Zuordnung aus mehreren Seiten bestehender Dateien gilt nicht für Tabellenbereiche für temporäre Tabellen.

Eine Tabelle besteht aus folgenden separaten Tabellenobjekten:

– Ein Datenobjekt. Hier werden die regulären Spaltendaten gespeichert.
– Ein Indexobjekt. Hier werden alle für die Tabelle definierten Indizes gespeichert.
– Ein Langfelddatenobjekt (LF-Datenobjekt). Hier werden Langfelddaten gespeichert, wenn die Tabelle eine oder mehrere Spalten mit LONG-Datentypen besitzt.
– Zwei LOB-Datenobjekte. Wenn die Tabelle eine oder mehrere LOB-Spalten hat, werden diese in folgenden beiden Tabellenobjekten gespeichert:
  - Ein Tabellenobjekt für die LOB-Daten
  - Ein zweites Tabellenobjekt für Metadaten, die die LOB-Daten beschreiben
– Ein Blockzuordnungsobjekt für MDC-Tabellen (MDC = mehrdimensionales Clustering)
– Ein zusätzliches XDA-Objekt, in dem XML-Dokumente gespeichert werden.

Jedes Tabellenobjekt wird getrennt gespeichert und ordnet nach Bedarf neue (durch EXTENTSIZE definierte) Bereiche zu. Jedes DMS-Tabellenobjekt wird außerdem mit einem Metadatenobjekt verbunden, das als Speicherbereichsmaske bezeichnet wird und alle EXTENTSIZE großen Bereiche im Tabellenbereich beschreibt, die zu dem Tabellenobjekt gehören. Der Speicherbereich für Speicherbereichsmasken wird ebenfalls jeweils in der Größe von EXTENTSIZE zugeordnet. Daher beträgt die Anfangszuordnung von Speicherbereich für ein Objekt in einem DMS-Tabellenbereich zwei EXTENTSIZE-Größen. (Die Anfangszuordnung von Speicherbereich für ein Objekt in einem SMS-Tabellenbereich beträgt eine Seite).

Wenn Sie viele kleine Tabellen in einem DMS-Tabellenbereich haben, ist es möglich, dass eine relativ große Menge an Speicher für eine relativ kleine Menge von Daten zugeordnet ist. Definieren Sie in einem solchen Fall einen kleinen Wert für EXTENTSIZE. Wenn Sie jedoch eine sehr umfangreiche Tabelle mit einer hohen Zuwachsrate haben und einen DMS-Tabellenbereich mit einem niedrigen Wert für EXTENTSIZE verwenden, könnte es passieren, dass häufiger zusätzliche Speicherbereiche zugeordnet werden, ohne dass ein Bedarf besteht.

• Art des Zugriffs auf die Tabellen

  Wenn auf die Tabellen durch zahlreiche Abfragen oder Transaktionen zugegriffen wird, die große Datenmengen verarbeiten, kann das Vorablesen von Daten aus den Tabellen eine wesentliche Leistungsverbesserung bewirken.

• Minimal erforderliche Anzahl von EXTENTSIZE-Speicherbereichen

  Falls nicht genügend Platz für fünf EXTENTSIZE-Speicherbereiche des Tabellenbereichs in den Containern vorhanden ist, wird der Tabellenbereich nicht erstellt.

Seitengröße, Tabellengröße und Tabellenbereichsgröße

Für DMS-Tabellenbereiche, DMS-Tabellenbereiche für temporäre Tabellen und Tabellenbereiche mit dynamischem Speicher für nicht temporäre Tabellen legt die Seitengröße, die Sie für Ihre Datenbank auswählen, die obere Begrenzung für die Tabellenbereichsgröße fest. Für Tabellen in SMS-Tabellenbereichen und Tabellenbereichen mit dynamischem Speicher für temporäre Tabellen, begrenzt die Seitengröße die Größe der Tabellen selbst.

Sie können eine Seitengrößenbegrenzung von 4, 8, 16 oder 32 KB verwenden. Für jede dieser Seitengrößen gelten zudem Maximalgrößen für die einzelnen Tabellenbereichstypen, die Sie einhalten müssen.

Tabelle 14 bietet eine Übersicht über die Begrenzungen von Tabellenbereichsgrößen für DMS-Tabellenbereiche und Tabellenbereiche mit dynamischem Speicher für nicht temporäre Tabellen nach Tabellengröße:

Tabelle 14. Größenbegrenzungen für DMS-Tabellenbereiche und Tabellenbereiche mit dynamischem Speicher für nicht temporäre Tabellen. DMS-Tabellenbereiche und Tabellenbereiche mit dynamischem Speicher für nicht temporäre Tabellen werden durch die Seitengröße beschränkt.

Tabellenbereichstyp	Seitengrößenbegrenzung - 4 KB	Seitengrößenbegrenzung - 8 KB	Seitengrößenbegrenzung - 16 KB	Seitengrößenbegrenzung - 32 KB
DMS-Tabellenbereiche und (reguläre) Tabellenbereiche mit dynamischem Speicher für nicht temporäre Tabellen	64 G	128 G	256 G	512 G
DMS-Tabellenbereiche, temporäre DMS-Tabellenbereiche und große Tabellenbereiche mit dynamischem Speicher für nicht temporäre Tabellen (Typ LARGE)	8192 G	16.384 G	32.768 G	65.536 G

Tabelle 15 bietet eine Übersicht über die Begrenzungen von Tabellenbereichsgrößen für SMS-Tabellenbereiche und Tabellenbereiche mit dynamischem Speicher für temporäre Tabellen nach Seitengröße:

Tabelle 15. Größenbegrenzungen für Tabellen in SMS-Tabellenbereichen und Tabellenbereichen mit dynamischem Speicher für temporäre Tabellen. Bei Tabellen in SMS-Tabellenbereichen und Tabellenbereichen mit dynamischem Speicher für temporäre Tabellen werden die Tabellenobjekte selbst und nicht die Tabellenbereiche durch die Seitengröße beschränkt.

Tabellenbereichstyp	Seitengrößenbegrenzung - 4 KB	Seitengrößenbegrenzung - 8 KB	Seitengrößenbegrenzung - 16 KB	Seitengrößenbegrenzung - 32 KB
SMS-Tabellenbereiche	64 G	128 G	256 G	512 G
SMS-Tabellenbereiche für temporäre Tabellen und Tabellenbereiche mit dynamischem Speicher für temporäre Tabellen	8192 G	16.384 G	32.768 G	65.536 G

Informationen zu Datenbank- und Indexseitengrößenbegrenzungen für die verschiedenen Typen von Tabellenbereichen finden Sie unter den seitengrößenspezifischen Begrenzungen des Datenbankmanagers in „SQL- und XML-Begrenzungen" (SQL and XML limits) im Handbuch *SQL Reference*.

Effizienz der Platten-E/A und Tabellenbereichsentwurf

Die Art und der Aufbau Ihres Tabellenbereichs bestimmen die Effizienz der Ein-/Ausgabeoperationen, die mit diesem Tabellenbereich erzielt werden kann.

Sie sollten sich mit den folgenden Konzepten vertraut machen, bevor Sie weitere Themen zum Entwerfen und Verwenden von Tabellenbereichen in Betracht ziehen:

Lesen großer Blöcke (Big Blocks)
Eine Leseoperation, bei der mehrere Seiten (in der Regel ein EXTENTSIZE

großer Bereich) in einer einzigen Anforderung abgerufen werden. Das Lesen mehrerer Seiten in einem Vorgang ist effizienter als das Lesen jeder Seite in getrennten Vorgängen.

Vorablesezugriff
Das Vorablesen der Seiten, auf die in einer Abfrage zugegriffen wird. Der Hauptzweck des Vorablesens ist die Verringerung von Antwortzeiten. Dies kann erreicht werden, wenn das Vorablesen von Seiten asynchron zur Ausführung der Abfrage stattfinden kann. Die besten Antwortzeiten werden erzielt, wenn entweder die CPU(s) oder das E/A-Subsystem mit maximaler Kapazität arbeiten.

Seitenlöschfunktion
Durch das Lesen und Ändern von Seiten sammeln diese sich im Pufferpool der Datenbank an. Wenn eine Datenseite eingelesen wird, wird sie in eine Seite des Pufferpools eingelesen. Wenn der Pufferpool ganz mit geänderten Seiten gefüllt ist, muss eine dieser geänderten Seiten auf die Platte geschrieben werden, bevor die neue Seite eingelesen werden kann. Bevor nun der Pufferpool gänzlich gefüllt wird, treten Seitenlöschagenten in Aktion, die geänderte Seiten auf die Platte schreiben und im Pufferpool löschen, um die Verfügbarkeit von Pufferpoolseiten für zukünftige Leseanforderungen sicherzustellen.

Immer, wenn der Datenbankmanager das Lesen großer Blöcke (Big Blocks) als vorteilhaft erkennt, werden Lesezugriffe in großen Blöcken ausgeführt. Dies tritt in der Regel beim Abrufen von Daten, die sequenzieller oder teilweise sequenzieller Art sind. Die Menge der Daten, die in einer Leseoperation gelesen werden, hängt von der Größe des Werts für EXTENTSIZE ab. Je größer der Wert für EXTENTSIZE ist, desto mehr Seiten können in einem Vorgang gelesen werden.

Die Leistung sequenzieller Vorableseoperationen kann weiterhin verbessert werden, wenn Seiten vom Datenträger in zusammenhängende Seiten im Pufferpool gelesen werden können. Da Pufferpools standardmäßig seitenbasiert sind, gibt es keine Garantie, dass sich beim Lesen vom Datenträger in zusammenhängende Seiten eine zusammenhängende Gruppe von Seiten finden lässt. Zu diesem Zweck können blockbasierte Pufferpools verwendet werden, weil diese nicht nur einen Seitenbereich, sondern außerdem einen Blockbereich für Gruppen zusammenhängender Seiten enthalten. Jede Gruppe zusammenhängender Seiten wird als Block bezeichnet, und jeder Block enthält eine Anzahl von Seiten, die als Blockgröße bezeichnet wird. Die Größen für den Seiten- und den Blockbereich sowie die Anzahl Seiten in jedem Block sind konfigurierbar.

Die Art, wie der Bereich auf der Platte gespeichert ist, hat Einfluss auf die E/A-Effizienz. In einem DMS-Tabellenbereich mit Einheitencontainern werden die Daten eher zusammenhängend auf der Platte gespeichert und können mit minimaler Suchzeit und Plattenlatenzzeit gelesen werden. Bei der Verwendung von Dateien führt eine große Datei, die zur Verwendung durch einen DMS-Tabellenbereich vorab zugeordnet wurde, ebenfalls eher dazu, dass die Datei auf der Platte zusammenhängend gespeichert wird, insbesondere wenn die Datei in einem noch ungenutzten Speicherbereich zugeordnet wurde. Die Daten können vom Dateisystem jedoch in Teile getrennt an mehr als einer Position auf der Platte gespeichert werden. Dies geschieht häufiger bei Verwendung von SMS-Tabellenbereichen, bei denen Dateien um jeweils eine Seite erweitert werden, wodurch die Fragmentierung wahrscheinlicher wird.

Sie können den Grad des Vorablesens durch Ändern des Parameters PREFETCHSIZE in der Anweisung CREATE TABLESPACE bzw. ALTER TABLESPACE steuern,

oder Sie können die Vorablesezugriffsgröße auf AUTOMATIC setzen, damit der Datenbankmanager automatisch die optimale Größe zur Verwendung auswählt. (Der Standardwert für alle Tabellenbereiche in der Datenbank wird durch den Konfigurationsparameter **dft_prefetch_sz** der Datenbank festgelegt.) Der Parameter PREFETCHSIZE gibt dem Datenbankmanager an, wie viele Seiten zu lesen sind, wenn ein Vorablesezugriff ausgelöst wird. Wenn der Wert des Parameters PREFETCHSIZE auf ein Vielfaches des Parameters EXTENTSIZE in der Anweisung CREATE TABLESPACE gesetzt wird, können mehrere EXTENTSIZE große Bereiche parallel gelesen werden. (Der Standardwert für alle Tabellenbereiche in der Datenbank wird durch den Konfigurationsparameter **dft_extent_sz** der Datenbank festgelegt.) Der Parameter EXTENTSIZE gibt die Anzahl der 4-KB-Seiten an, die in einen Container geschrieben werden, bevor zum nächsten Container übergegangen wird.

Nehmen Sie zum Beispiel an, Sie hätten einen Tabellenbereich, der drei Einheiten verwendet. Wenn Sie den Wert für PREFETCHSIZE auf das Dreifache des Werts für EXTENTSIZE setzen, kann der Datenbankmanager einen Lesezugriff in großen Blöcken von jeder Einheit parallel durchführen, wodurch der E/A-Durchsatz erheblich erhöht wird. Voraussetzungen sind, dass jede Einheit eine getrennte physische Einheit ist und dass der Controller über eine ausreichende Bandbreite verfügt, um den Datenstrom von jeder Einheit zu verarbeiten. Beachten Sie, dass der Datenbankmanager eventuell die Parameter für den Vorablesezugriff zur Laufzeit aufgrund der Abfragegeschwindigkeit, der Pufferpoolauslastung und anderer Faktoren dynamisch anpassen muss.

Einige Dateisysteme (z. B. Journaled File System unter AIX) verfügen über eine eigene Vorablesemethode. In einigen Fällen kann der Vorablesezugriff des Dateisystems auf größere Datenmengen eingestellt sein als der Vorablesezugriff des Datenbankmanagers. Dies kann dazu führen, dass der Vorablesezugriff für SMS- und DMS-Tabellenbereiche mit Dateicontainern scheinbar eine bessere Leistung zeigt als der Vorablesezugriff für DMS-Tabellenbereiche mit Einheitencontainern. Dies ist jedoch irreführend, da es sich wahrscheinlich um das Ergebnis einer zusätzlichen Ebene des Vorablesezugriffs handelt, die innerhalb des Dateisystems wirksam ist. Normalerweise sollten DMS-Tabellenbereiche jeder äquivalenten Konfiguration im Hinblick auf die Leistung überlegen sein.

Für ein effizientes Vorablesen (oder auch nur Lesen) muss eine ausreichende Anzahl verfügbarer Pufferpoolseiten vorhanden sein. Zum Beispiel könnte es eine Anforderung zum parallelen Vorablesezugriff geben, mit dem drei (durch EXTENTSIZE bestimmte) Bereiche aus einem Tabellenbereich gelesen werden und durch den für jede einzulesende Seite eine geänderte Seite aus dem Pufferpool herausgeschrieben wird. Die Anforderung zum Vorablesezugriff könnte so weit verlangsamt werden, dass sie mit der Abfrage nicht Schritt halten kann. Daher sollten Seitenlöschfunktionen in ausreichender Anzahl konfiguriert werden, um die Anforderungen von Vorablesezugriffen erfüllen zu können.

Erstellen von Tabellenbereichen

Durch das Erstellen eines Tabellenbereichs innerhalb einer Datenbank werden dem Tabellenbereich Container zugeordnet und die zugehörigen Definitionen und Attribute im Datenbanksystemkatalog gespeichert.

Informationen zu diesem Vorgang

Bei Tabellenbereichen mit dynamischem Speicher ordnet der Datenbankmanager einem Tabellenbereich Container auf der Basis der der Datenbank zugeordneten Speicherpfade zu.

Bei Tabellenbereichen ohne dynamischen Speicher müssen Sie die Pfad-, Einheiten- oder Dateinamen für die Container kennen, die Sie bei der Erstellung Ihrer Tabellenbereiche verwenden wollen. Darüber hinaus müssen Sie für jeden Einheiten- oder Dateicontainer, den Sie für DMS-Tabellenbereiche erstellen, wissen, wie viel Speicherkapazität Sie den einzelnen Containern zuordnen können.

Vorgehensweise

- Geben Sie in die Befehlszeile die folgende Anweisung ein, um einen SMS-Tabellenbereich zu erstellen:

  ```
  CREATE TABLESPACE name
        MANAGED BY SYSTEM
        USING ('pfad')
  ```

- Geben Sie in die Befehlszeile die folgende Anweisung ein, um einen DMS-Tabellenbereich zu erstellen:

  ```
  CREATE TABLESPACE name
        MANAGED BY DATABASE
        USING (FILE'pfad' größe)
  ```

  Beachten Sie, dass DMS-Tabellenbereiche standardmäßig als große Tabellenbereiche erstellt werden.

- Geben Sie in die Befehlszeile eine der folgenden Anweisungen ein, um einen Tabellenbereich mit dynamischem Speicher zu erstellen:

  ```
  CREATE TABLESPACE  name
  ```

  oder

  ```
  CREATE TABLESPACE name
        MANAGED BY AUTOMATIC STORAGE
  ```

  Unter der Annahme, dass der Tabellenbereich in einer Datenbank mit dynamischem Speicher erstellt wird, sind die beiden oben gezeigten Anweisungen äquivalent. Tabellenbereiche, die in einer solchen Datenbank erstellt werden, werden standardmäßig als Tabellenbereiche mit dynamischem Speicher erstellt, sofern Sie nichts anderes angeben.

Beispiel

Beispiel 1: Erstellen eines SMS-Tabellenbereichs unter Windows. Mit der folgenden SQL-Anweisung wird ein SMS-Tabellenbereich mit dem Namen RESOURCE mit Containern in drei Verzeichnissen auf drei separaten Laufwerken erstellt:

```
CREATE TABLESPACE RESOURCE
      MANAGED BY SYSTEM
      USING ('d:\acc_tbsp', 'e:\acc_tbsp', 'f:\acc_tbsp')
```

Beispiel 2: Erstellen eines DMS-Tabellenbereichs unter Windows. Mit der folgenden SQL-Anweisung wird ein DMS-Tabellenbereich mit zwei Dateicontainern von je 5000 Seiten erstellt:

```
CREATE TABLESPACE RESOURCE
      MANAGED BY DATABASE
      USING (FILE'd:\db2data\acc_tbsp' 5000,
             FILE'e:\db2data\acc_tbsp' 5000)
```

In den beiden gezeigten Beispielen werden explizit Namen für die Container angegeben. Wenn Sie jedoch relative Containernamen angeben, wird der Container in dem für die Datenbank angelegten Unterverzeichnis erstellt.

Bei der Erstellung von Tabellenbereichscontainern erstellt der Datenbankmanager alle Verzeichnisebenen, die noch nicht vorhanden sind. Wenn zum Beispiel ein Container als `/project/user_data/container1` angegeben wird und das Verzeichnis `/project` nicht vorhanden ist, erstellt der Datenbankmanager die Verzeichnisse `/project` und `/project/user_data`.

Alle vom Datenbankmanager erstellten Verzeichnisse werden mit PERMISSION 711 erstellt. PERMISSION 711 ist für den Zugriff auf abgeschirmte Prozesse erforderlich. Das bedeutet, dass der Instanzeigner über Lese-, Schreib- und Ausführungszugriff verfügt, die übrigen Benutzer verfügen über Ausführungszugriff. Benutzer mit Ausführungszugriff sind auch berechtigt, die Verzeichnisse der Tabellenbereichscontainer zu durchsuchen. Da nur der Instanzeigner über Lese- und Schreibzugriff verfügt, kann das folgende Szenario auftreten, wenn mehrere Instanzen erstellt werden:

- Nehmen Sie unter Verwendung derselben Verzeichnisstruktur wie im obigen Beispiel an, dass die Verzeichnisebenen `/project/user_data` nicht vorhanden sind.
- Benutzer 'user1' erstellt eine Instanz, die standardmäßig den Namen 'user1' erhält. Anschließend erstellt dieser Benutzer eine Datenbank und einen Tabellenbereich mit `/project/user_data/container1` als einem seiner Container.
- Benutzer 'user2' erstellt eine Instanz mit dem Standardnamen 'user2'. Anschließend versucht er, einen Tabellenbereich mit `/project/user_data/container2` als einem seiner Container zu erstellen.

Da der Datenbankmanager die Verzeichnisebenen `/project/user_data` mit PERMISSION 700 aus der ersten Anforderung erstellt hat, besitzt Benutzer 'user2' keinen Zugriff auf diese Verzeichnisebenen und kann 'container2' in diesen Verzeichnissen nicht erstellen. In diesem Fall schlägt die Operation CREATE TABLESPACE fehl.

Dieser Konflikt lässt sich mit zwei Methoden lösen:

1. Erstellen Sie das Verzeichnis `/project/user_data`, bevor Sie die Tabellenbereiche erstellen, und erteilen Sie die Zugriffsberechtigungen, die für beide Benutzer ('user1' und 'user2') zur Erstellung der Tabellenbereiche erforderlich sind. Wenn alle Ebenen des Tabellenbereichsverzeichnisses vorhanden sind, modifiziert der Datenbankmanager die Zugriffsberechtigungen nicht.
2. Wenn Benutzer 'user1' den Tabellenbereichscontainer `/project/user_data/container1` erstellt hat, stellen Sie die Zugriffsberechtigungen für `/project/user_data` ein, die Benutzer 'user2' zur Erstellung des Tabellenbereichs benötigt.

Wird ein Unterverzeichnis vom Datenbankmanager erstellt, kann es auch wieder vom Datenbankmanager gelöscht werden, wenn der Tabellenbereich gelöscht wird.

In diesem Szenario wird davon ausgegangen, dass die Tabellenbereiche nicht einer bestimmten Datenbankpartitionsgruppe zugeordnet werden. Die Standarddatenbankpartitionsgruppe IBMDEFAULTGROUP wird verwendet, wenn der folgende Parameter in der Anweisung nicht angegeben wird:

`IN` *`datenbankpartitionsgruppenname`*

Beispiel 3: Erstellen von DMS-Tabellenbereichen unter AIX. Mit der folgenden SQL-Anweisung werden ein DMS-Tabellenbereich auf einem AIX-System mit drei logischen Datenträgern von je 10.000 Seiten erstellt und die zugehörigen E/A-Merkmale angegeben:

```
CREATE TABLESPACE RESOURCE
      MANAGED BY DATABASE
      USING (DEVICE '/dev/rdblv6' 10000,
             DEVICE '/dev/rdblv7' 10000,
             DEVICE '/dev/rdblv8' 10000)
      OVERHEAD 7.5
      TRANSFERRATE 0.06
```

Die in der SQL-Anweisung angegebenen UNIX-Einheiten müssen bereits vorhanden sein, und der Instanzeigner und die SYSADM-Gruppe müssen Schreibzugriff auf sie haben.

Beispiel 4: Erstellen eines DMS-Tabellenbereichs auf einem UNIX-System. Im folgenden Beispiel wird ein DMS-Tabellenbereich in einer Datenbankpartitionsgruppe mit dem Namen ODDGROUP in einer UNIX-Datenbank mit mehreren Datenbankpartitionen erstellt. Die Datenbankpartitionsgruppe ODDGROUP muss zuvor mit der Anweisung CREATE DATABASE PARTITION GROUP erstellt worden sein. Im vorliegenden Beispiel wird angenommen, dass die Datenbankpartitionsgruppe ODDGROUP aus den Datenbankpartitionen 1, 3 und 5 besteht. In allen Datenbankpartitionen ist die Einheit `/dev/hdisk0` für 10.000 4-KB-Seiten zu verwenden. Außerdem wird für jede Datenbankpartition eine Einheit von 40.000 4-KB-Seiten deklariert.

```
CREATE TABLESPACE PLANS IN ODDGROUP
       MANAGED BY DATABASE
       USING (DEVICE '/dev/HDISK0' 10000, DEVICE '/dev/n1hd01' 40000)
             ON DBPARTITIONNUM 1
            (DEVICE '/dev/HDISK0' 10000, DEVICE '/dev/n3hd03' 40000)
             ON DBPARTITIONNUM 3
            (DEVICE '/dev/HDISK0' 10000, DEVICE '/dev/n5hd05' 40000)
             ON DBPARTITIONNUM 5
```

Der Datenbankmanager kann die Leistung sequenzieller E/A-Operationen durch die Verwendung des sequenziellen Vorablesezugriffs (Sequential Prefetching) wesentlich verbessern, da dieser mit parallelen E/A-Operationen arbeitet.

Beispiel 5: Erstellen eines SMS-Tabellenbereichs mit einer größeren Seitengröße als der Standardgröße. Sie können auch einen Tabellenbereich erstellen, der den Standardwert von 4 KB für die Seitengröße übersteigt. Durch die folgende SQL-Anweisung wird ein SMS-Tabellenbereich auf einem Linux- und UNIX-System mit einer Seitengröße von 8 KB erstellt:

```
CREATE TABLESPACE SMS8K
      PAGESIZE 8192
      MANAGED BY SYSTEM
      USING ('FSMS_8K_1')
      BUFFERPOOL BUFFPOOL8K
```

Beachten Sie, dass der zugeordnete Pufferpool ebenfalls die Seitengröße von 8 KB aufweisen muss.

Der erstellte Tabellenbereich kann nicht verwendet werden, bis der Pufferpool, auf den er verweist, aktiviert ist.

Die Anweisung ALTER TABLESPACE kann dazu verwendet werden, Container in einem DMS-Tabellenbereich hinzuzufügen, zu löschen oder in der Größe zu än-

dern sowie die Einstellungen für die Parameter PREFETCHSIZE, OVERHEAD und TRANSFERRATE für einen Tabellenbereich zu modifizieren. Die Transaktion, die die Anweisung für den Tabellenbereich absetzt, sollte möglichst bald nach der SQL-Anweisung ALTER TABLESPACE festgeschrieben werden, um Konkurrenzsituationen beim Zugriff auf den Systemkatalog zu vermeiden.

Anmerkung: Für den Wert für PREFETCHSIZE sollte ein Vielfaches des Werts für EXTENTSIZE verwendet werden. Wenn beispielsweise für EXTENTSIZE der Wert 10 angegeben wird, sollte der Wert für PREFETCHSIZE 20 oder 30 sein. Sie sollten zudem in Betracht ziehen, den Wert für PREFETCHSIZE vom Datenbankmanager automatisch bestimmen zu lassen, indem Sie PREFETCHSIZE auf AUTOMATIC setzen.

Eine direkte Ein-/Ausgabe (Direct I/O, DIO) verbessert die Speicherleistung, weil das Caching auf der Dateisystemebene umgangen wird. Dieser Prozess verringert den CPU-Aufwand und stellt der Datenbankinstanz mehr Speicher zur Verfügung.

Eine gleichzeitige Ein-/Ausgabe (Concurrent I/O, CIO) bietet die Vorteile von DIO und entlastet zudem die serielle Verarbeitung von Schreibzugriffen.

DIO und CIO werden von AIX unterstützt. DIO wird von HP-UX, Solaris, Linux und Windows-Betriebssystemen unterstützt.

Die Schlüsselwörter NO FILE SYSTEM CACHING und FILE SYSTEM CACHING sind Teil der SQL-Anweisungen CREATE TABLESPACE und ALTER TABLESPACE und ermöglichen die Angabe, ob DIO oder CIO für die einzelnen Tabellenbereiche zu verwenden ist. Wenn NO FILE SYSTEM CACHING definiert ist, versucht der Datenbankmanager, nach Möglichkeit gleichzeitige Ein-/Ausgabeoperationen (Concurrent I/O, CIO) zu nutzen. In Fällen, in denen CIO nicht unterstützt wird (z. B. bei Verwendung von JFS), wird stattdessen DIO verwendet.

Wenn Sie die Anweisung CREATE TABLESPACE absetzen, wird die Funktion zur Recovery gelöschter Tabellen standardmäßig aktiviert. Diese Funktion ermöglicht es, gelöschte Tabellendaten über einen Restore und anschließende aktualisierende Recovery auf Tabellenbereichsebene wiederherzustellen. Diese Art der Recovery ist hilfreich, da sie weniger Zeit in Anspruch nimmt als eine Recovery auf Datenbankebene; zudem bleibt Ihre Datenbank für die Benutzer verfügbar.

Allerdings kann sich die Funktion zur Recovery gelöschter Tabellen auf die Leistung der aktualisierenden Recovery auswirken, wenn viele Tabellenlöschoperationen wiederhergestellt werden sollen oder wenn die Protokolldatei sehr groß ist.

Sie können diese Funktion inaktivieren, wenn Sie zahlreiche Tabellenlöschoperationen ausführen möchten und Sie entweder die Umlaufprotokollierung verwenden oder nicht beabsichtigen, eine der gelöschten Tabellen wiederherzustellen. Zur Inaktivierung dieser Funktion können Sie die Option DROPPED TABLE RECOVERY explizit auf OFF setzen, wenn Sie die Anweisung CREATE TABLESPACE absetzen. Alternativ dazu können Sie die Funktion zur Recovery gelöschter Tabellen für einen vorhandenen Tabellenbereich mit der Anweisung ALTER TABLESPACE inaktivieren.

Erstellen von Tabellenbereichen für temporäre Tabellen

Tabellenbereiche für temporäre Tabellen enthalten temporäre Daten, die der Datenbankmanager zur Ausführung von Operationen wie Sortierungen und Joins benötigt, da für solche Operationen zusätzlicher Speicherplatz zur Verarbeitung der Er-

gebnismengen erforderlich ist. Sie können Tabellenbereiche für temporäre Tabellen mithilfe einer Variante der Anweisung CREATE TABLESPACE erstellen.

Informationen zu diesem Vorgang

Ein *Tabellenbereich für temporäre Systemtabellen* dient zum Speichern temporärer Systemtabellen. Eine Datenbank muss immer über mindestens einen Tabellenbereich für temporäre Systemtabellen verfügen, da temporäre Systemtabellen nur in einem solchen Tabellenbereich gespeichert werden können. Beim Erstellen einer Datenbank wird einer der drei Standardtabellenbereiche als Tabellenbereich für temporäre Systemtabellen mit dem Namen "TEMPSPACE1" definiert. Sie sollten mindestens einen Tabellenbereich für temporäre Systemtabellen für jede Seitengröße für die Tabellenbereiche für Benutzertabellen haben, die in Ihrer Datenbank enthalten sind. Ansonsten könnten einige Abfragen fehlschlagen. Weitere Informationen finden Sie in „Tabellenbereiche für Systemdaten, Benutzerdaten und temporäre Daten" auf Seite 167.

Tabellenbereiche für temporäre Benutzertabellen werden nicht standardmäßig erstellt, wenn eine Datenbank erstellt wird. Wenn für Ihre Anwendungsprogramme temporäre Tabellen erforderlich sind, müssen Sie einen Tabellenbereich für temporäre Benutzertabellen erstellen, in dem die temporären Tabellen gespeichert werden. Ebenso wie reguläre Tabellenbereiche können Tabellenbereiche für temporäre Benutzertabellen in allen Datenbankpartitionsgruppen mit Ausnahme von IBMTEMPGROUP erstellt werden. IBMDEFAULTGROUP ist die Standarddatenbankpartitionsgruppe, die beim Erstellen eines Tabellenbereichs für temporäre Benutzertabellen verwendet wird.

Einschränkungen

Für Tabellenbereiche für temporäre Systemtabellen in einer partitionierten Umgebung ist IBMTEMPGROUP die einzige Datenbankpartitionsgruppe, die bei der Erstellung solcher Tabellenbereiche angegeben werden kann.

Vorgehensweise

- Zur Erstellung eines Tabellenbereichs für temporäre Systemtabellen neben dem Standardtabellenbereich TEMPSPACE1 verwenden Sie eine Anweisung CREATE TABLESPACE mit den Schlüsselwörtern SYSTEM TEMPORARY. Beispiel:

```
CREATE SYSTEM TEMPORARY TABLESPACE tmp_tbsp
  MANAGED BY SYSTEM
  USING ('d:\tmp_tbsp','e:\tmp_tbsp')
```

- Zur Erstellung eines Tabellenbereichs für temporäre Benutzertabellen verwenden Sie die Anweisung CREATE TABLESPACE mit den Schlüsselwörtern USER TEMPORARY. Beispiel:

```
CREATE USER TEMPORARY TABLESPACE usr_tbsp
  MANAGED BY DATABASE
  USING (FILE 'd:\db2data\user_tbsp' 5000,
  FILE 'e:\db2data\user_tbsp' 5000)
```

Definieren der ersten Tabellenbereiche bei der Datenbankerstellung

Bei der Erstellung einer Datenbank werden drei Tabellenbereiche definiert: (1) SYSCATSPACE für die Systemkatalogtabellen, (2) TEMPSPACE1 für temporäre Systemtabellen, die während der Datenbankverarbeitung erstellt werden, und (3) USERSPACE1 für benutzerdefinierte Tabellen und Indizes. Darüber hinaus können Sie gleichzeitig weitere Tabellenbereiche für Benutzertabellen erstellen.

Informationen zu diesem Vorgang

Anmerkung: Wenn Sie eine Datenbank zum ersten Mal erstellen, wird für diese kein Tabellenbereich für temporäre Benutzertabellen erstellt.

Sofern nicht anders angegeben, werden die drei Standardtabellenbereiche durch den dynamischen Speicher verwaltet.

Bei Verwendung des Befehls **CREATE DATABASE** können Sie die Seitengröße für den Standardpufferpool und die ersten Tabellenbereiche angeben. Dieser Standardwert stellt auch die Standardseitengröße für alle zukünftigen Anweisungen CREATE BUFFERPOOL und CREATE TABLESPACE dar. Wenn Sie die Seitengröße beim Erstellen der Datenbank nicht angeben, wird die Standardseitengröße von 4 KB verwendet.

Geben Sie Folgendes in die Befehlszeile ein, um erste Tabellenbereiche zu definieren:

```
CREATE DATABASE name
  PAGESIZE seitengröße
  CATALOG TABLESPACE
    MANAGED BY SYSTEM USING ('pfad')
    EXTENTSIZE wert PREFETCHSIZE wert
  USER TABLESPACE
    MANAGED BY DATABASE USING (FILE'pfad' 5000,
                               FILE'pfad' 5000)
    EXTENTSIZE wert PREFETCHSIZE wert
  TEMPORARY TABLESPACE
    MANAGED BY SYSTEM USING ('pfad')
  WITH "kommentar"
```

Wenn Sie die Standarddefinition für diese Tabellenbereiche nicht verwenden möchten, können Sie die Merkmale im Befehl **CREATE DATABASE** angeben. Zum Beispiel könnte der folgende Befehl zur Erstellung der Datenbank unter Windows verwendet werden:

```
CREATE DATABASE PERSONL
  PAGESIZE 16384
  CATALOG TABLESPACE
    MANAGED BY SYSTEM USING ('d:\pcatalog','e:\pcatalog')
    EXTENTSIZE 16 PREFETCHSIZE 32
  USER TABLESPACE
    MANAGED BY DATABASE USING (FILE'd:\db2data\personl' 5000,
                               FILE'd:\db2data\personl' 5000)
    EXTENTSIZE 32 PREFETCHSIZE 64
  TEMPORARY TABLESPACE
    MANAGED BY SYSTEM USING ('f:\db2temp\personl')
  WITH "Personnel DB for BSchiefer Co"
```

In diesem Beispiel werden die Standardseitengröße auf den Wert 16.384 gesetzt und die Definitionen für jeden der ersten Tabellenbereiche explizit angegeben. Definitionen müssen nur für die Tabellenbereiche angegeben werden, für die die Standarddefinition nicht verwendet werden soll.

Anmerkung: In einer Umgebung mit partitionierten Datenbanken können Sie keine Container erstellen oder bestimmten Datenbankpartitionen zuordnen. Sie müssen die Datenbank zunächst mit den Standardtabellenbereichen für Benutzertabellen und temporäre Tabellen erstellen. Anschließend können Sie mithilfe der Anweisung CREATE TABLESPACE die erforderlichen Tabellenbereiche erstellen. Im letzten Schritt können Sie die Standardtabellenbereiche löschen.

Für die Codierung des Ausdrucks MANAGED BY im Befehl **CREATE DATABASE** gilt dasselbe Format wie für den Ausdruck MANAGED BY im Befehl CREATE TABLESPACE.

Sie können weitere Benutzertabellenbereiche und Tabellenbereiche für temporäre Tabellen nach Wunsch hinzufügen. Sie können den Katalogtabellenbereich SYSCATSPACE weder löschen noch einen anderen erstellen. Außerdem muss immer mindestens ein Tabellenbereich für temporäre Systemtabellen mit einer Seitengröße von 4 KB vorhanden sein. Sie können weitere Tabellenbereiche für temporäre Systemtabellen erstellen. Nach der Erstellung können auch die Seitengröße und der EXTENTSIZE-Wert eines Tabellenbereichs nicht mehr geändert werden.

Verbindung zu DMS-Einheiten mit direktem Zugriff herstellen

Wenn Container zum Speichern von Daten verwendet werden, unterstützt der Datenbankmanager einen direkten Plattenzugriff (unformatierte Ein-/Ausgabe).

Informationen zu diesem Vorgang

Diese Art der Unterstützung gibt Ihnen die Möglichkeit, eine Einheit mit direktem Plattenzugriff an ein DB2-Datenbanksystem anzuschließen.

Sie müssen die Einheiten- bzw. Dateinamen der Container kennen, auf die Sie bei der Erstellung Ihrer Tabellenbereiche verweisen wollen. Sie müssen wissen, wie viel Speicherplatz auf einer Einheit bzw. in einer Datei, die dem Tabellenbereich zugeordnet werden soll, zur Verfügung steht. Sie benötigen die richtigen Berechtigungen für den Schreib- und Lesezugriff auf den Container.

Die physischen und logischen Verfahren zum Angeben eines direkten Plattenzugriffs unterscheiden sich in Abhängigkeit vom Betriebssystem:

- Unter Windows-Betriebssystemen:

  Verwenden Sie die folgende Syntax, um ein physisches Festplattenlaufwerk anzugeben:

  ```
  \\.\PhysicalDriveN
  ```

  Dabei steht N für eines der physischen Laufwerke im System. Im vorliegenden Fall kann N durch die Werte 0, 1, 2 oder eine beliebige andere positive ganze Zahl ersetzt werden:

  ```
  \\.\PhysicalDrive5
  ```

  Verwenden Sie die folgende Syntax, um ein logisches Laufwerk (d. h. eine unformatierte Datenbankpartition) anzugeben:

  ```
  \\.\N:
  ```

  Dabei ist N: ein Buchstabe eines logischen Laufwerks im System. Die Angabe N: kann z. B. durch E: oder eine beliebige andere Angabe für den Laufwerkbuchstaben ersetzt werden. Um die Einschränkung zu überwinden, die durch die Verwendung eines Buchstabens zur Angabe des Laufwerks gegeben ist, können Sie eine global eindeutige ID (GUID) für das logische Laufwerk verwenden.

  Für Windows gibt es eine neue Methode zur Angabe unformatierter DMS-Tabellenbereichscontainer. Datenträgern (d. h. Datenbankpartitionen auf Basisfestplatten oder dynamischen Datenträgern) wird bei ihrer Erstellung eine global eindeutige ID (GUID) zugeordnet. Die GUID kann als Einheiten-ID bei der Angabe der Container in einer Tabellenbereichsdefinition verwendet werden. Die GUIDs sind über Systeme hinweg eindeutig. Das heißt, dass sie in einer Mehrpartitionsdatenbank für jede Datenbankpartition unterschiedlich sind, selbst wenn die Definitionen der Plattenpartitionen übereinstimmen.

Es steht ein Tool mit dem Namen *db2listvolumes.exe* (nur für Windows-Betriebssysteme) zur Verfügung, das die Anzeige der GUIDs für alle Plattendatenträger vereinfacht, die auf einem Windows-System definiert sind. Dieses Tool erstellt zwei Dateien in dem Verzeichnis, in dem es ausgeführt wird. Eine Datei mit dem Namen `volumes.xml` enthält in XML codierte Informationen über jeden Plattendatenträger, die einfach mit einem beliebigen XML-fähigen Browser angezeigt werden können. Die zweite Datei mit dem Namen `tablespace.ddl` enthält die erforderliche Syntax zur Angabe der Tabellenbereichscontainer. Diese Datei muss so aktualisiert werden, dass sie die noch ausstehenden Informationen angibt, die für eine Tabellenbereichsdefinition erforderlich sind. Der Befehl **db2listvolumes** benötigt keine Befehlszeilenargumente.

- Auf Linux- und UNIX-Plattformen kann ein logischer Datenträger Benutzern und Anwendungen als einzelne, zusammenhängende und erweiterbarer Plattendatenträger angezeigt werden. Trotzdem kann er sich auf nicht zusammenhängenden physischen Datenbankpartitionen und sogar mehr als einem physischen Datenträger befinden. Der logische Datenträger muss außerdem in einer einzigen Datenträgergruppe enthalten sein. Es besteht eine Begrenzung auf 256 logische Datenträger pro Datenträgergruppe. Es besteht eine Begrenzung auf 32 physische Datenträger pro Datenträgergruppe. Sie können mithilfe des Befehls **mklv** zusätzliche logische Datenträger erstellen. Mithilfe dieses Befehls können Sie den Namen des logischen Datenträgers angeben und seine Merkmale einschließlich der Anzahl und der Position der logischen Partitionen definieren, die für ihn zugeordnet werden sollen.

  Nachdem Sie einen logischen Datenträger erstellt haben, können Sie seinen Namen und seine Merkmale mithilfe des Befehls **chlv** ändern sowie die Anzahl der ihm zugeordneten logischen Partitionen mit dem Befehl **extendlv** erhöhen. Die standardmäßig vorgegebene Maximalgröße für einen logischen Datenträger bei der Erstellung sind 512 logische Partitionen, sofern sie nicht größer angegeben wurde. Mithilfe des Befehls **chlv** kann diese Begrenzung außer Kraft gesetzt werden.

  In AIX wird die Gruppe der Betriebssystembefehle, Bibliothekssubroutinen und anderen Tools, mit denen Sie logischen Datenträgerspeicher aufbauen und steuern können, Logical Volume Manager (LVM) genannt. LVM steuert Datenträgerressourcen, indem Daten zwischen einer einfacheren und flexiblen logischen Sicht des Speicherbereichs und den tatsächlichen physischen Datenträgern zugeordnet werden.

  Weitere Informationen zu **mklv** und anderen Befehlen für logische Datenträger sowie zu LVM finden Sie im Handbuch *AIX 5L Version 5.2 System Management Concepts: Operating System and Devices*.

Konfigurieren und Einrichten des direkten DMS-Plattenzugriffs (Linux)

Wenn Container zum Speichern von Daten verwendet werden, unterstützt der Datenbankmanager einen direkten Plattenzugriff (auf Roheinheiten) über die Blockeinheitenschnittstelle (d. h. unformatierte Ein-/Ausgabe - Raw I/O).

Vorbereitende Schritte

Bevor Sie eine unformatierte Ein-/Ausgabe unter Linux einrichten können, ist mindestens eine freie Datenbankpartition auf IDE- oder SCSI-Platten erforderlich. Um die Plattenpartition beim Erstellen des Tabellenbereichs angeben zu können, müssen Sie wissen, welchen Namen die Plattenpartition hat und wie viel Speicherplatz der Plattenpartition zugeordnet ist, die dem Tabellenbereich zugeordnet werden soll.

Informationen zu diesem Vorgang

Vor Version 9 wurde ein direkter Plattenzugriff über ein Controllerdienstprogramm für unformatierte Ein-/Ausgabe (Raw-Controller) unter Linux verwendet. Diese Methode ist jetzt veraltet, und von ihrer Verwendung wird abgeraten. Der Datenbankmanager ermöglicht die Verwendung dieser Methode, sofern das Betriebssystem Linux sie noch unterstützt. Jedoch wird in die Datei **db2diag** in diesem Fall eine Nachricht geschrieben, die darauf hinweist, dass diese Methode veraltet ist.

Bei der früheren Methode mussten Sie eine Plattenpartition an einen Raw-Controller "binden" und anschließend diesen Raw-Controller mit Hilfe des Befehls CREATE TABLESPACE für den Datenbankmanager definieren:

```
CREATE TABLESPACE dms1
MANAGED BY DATABASE
USING (DEVICE '/dev/raw/raw1' 1170736)
```

In einer Linux-Umgebung sind dabei die folgenden Informationen zu beachten. Unter Linux/390 unterstützt der Datenbankmanager Einheiten mit direktem Plattenzugriff nicht.

Vorgehensweise

Gehen Sie wie folgt vor, um die unformatierte Ein-/Ausgabe unter Linux zu konfigurieren:

1. Berechnen Sie die Anzahl der 4.096 Byte großen Seiten in dieser Datenbankpartition, und runden Sie gegebenenfalls ab. Beispiel:

   ```
   # fdisk /dev/sda
   Command (m for help): p

   Disk /dev/sda: 255 heads, 63 sectors, 1106 cylinders
   Units = cylinders of 16065 * 512 bytes
   ```

   In diesem Beispiel ist /dev/sda5 die zu verwendende unformatierte Datenbankpartition. Diese Partition sollte keine wertvollen Daten enthalten.

   Tabelle 16. Berechnungen für unformatierte E/A unter Linux

Einheitenstart	Anfang	Ende	Blöcke	ID	System
/dev/sda1	1	523	4200997	83	Linux
/dev/sda2	524	1106	4682947+	5	Extended
/dev/sda5	524	1106	4682947	83	Linux

   ```
   Command (m for help): q
   #
   ```

   Die Anzahl der Seiten in /dev/sda5 ist:

   ```
   num_pages = floor( (4682947 * 1024)/4096 )
   num_pages = 1170736
   ```

2. Erstellen Sie den Tabellenbereich, indem Sie den Namen der Plattenpartition angeben. Beispiel:

   ```
   CREATE TABLESPACE dms1
   MANAGED BY DATABASE
   USING (DEVICE '/dev/sda5' 1170736)
   ```

3. Wenn Sie logische Partitionen mit Junctionpunkten (oder Datenträgermountpunkten) angeben möchten, müssen Sie die Rohpartition an einen anderen

NTFS-formatierten Datenträger als Junctionpunkt anhängen und anschließend den Pfad zu dem Junctionpunkt auf dem NTFS-Datenträger als Containerpfad angeben. Beispiel:

```
CREATE TABLESPACE TS4
   MANAGED BY DATABASE USING (DEVICE 'C:\JUNCTION\DISK_1' 10000,
      DEVICE 'C:\JUNCTION\DISK_2' 10000)
```

Der Datenbankmanager fragt zuerst die Partition ab, um festzustellen, ob ein Dateisystem vorhanden ist. Ist dies der Fall, wird die Partition nicht als Roheinheit (RAW) betrachtet, und der Datenbankmanager führt in der Partition normale Dateisystem-E/A-Operationen aus.

Tabellenbereiche auf Roheinheiten werden auch für alle anderen Seitengrößen unterstützt, die vom Datenbankmanager unterstützt werden.

Ändern von Tabellenbereichen

Zur Änderung eines Tabellenbereichs über die Befehlszeile verwenden Sie die Anweisung ALTER TABLESPACE.

Informationen zu diesem Vorgang

Abhängig vom Typ des Tabellenbereichs haben Sie folgende Möglichkeiten:

- Vergrößern des Tabellenbereichs durch Hinzufügen zusätzlicher Container
- Ändern der Größe vorhandener Container
- Löschen von Containern
- Neuverteilen der Daten des Tabellenbereichs, um mit der Verwendung neuer Container zu beginnen oder Daten aus gelöschten Containern in andere Container zu versetzen
- Senken der oberen Grenze für den Tabellenbereich
- Verringern der Gesamtgröße des Tabellenbereichs

Sie können einen Tabellenbereich darüber hinaus umbenennen und ihn aus dem Offlinemodus in den Onlinemodus versetzen.

Berechnung der Tabellenbereichsbelegung

Mithilfe der Tabellenfunktion MON_GET_TABLESPACE können Sie ermitteln, wie viel Ihres Tabellenbereichs zurzeit belegt ist. Die Informationen, die von dieser Funktion zurückgegeben werden, helfen Ihnen bei der Entscheidung, ob nicht belegter Speicherbereich wieder freigegeben werden sollte.

Informationen zu diesem Vorgang

Diese Task stellt die Informationen bereit, mit deren Hilfe Sie das Ausmaß ermitteln können, bis zu dem ungenutzter Speicherbereich unterhalb der oberen Grenze für Ihren Tabellenbereich vorhanden ist. Auf der Grundlage dieser Informationen können Sie entscheiden, ob es vorteilhaft ist, ungenutzten Speicherbereich freizugeben.

Einschränkungen

Obwohl Sie verschiedene Nutzungsattribute zu allen Ihren Tabellenbereichen ermitteln können, verfügen nur Tabellenbereiche, die mit DB2 Version 9.7 oder einer späteren Version erstellt wurden, über die Funktion für konsolidierbaren Speicher. Wenn Sie in der Lage sein wollen, Speicher in Tabellenbereichen freizugeben, die

mit früheren Versionen des DB2-Datenbankprodukts erstellt wurden, müssen Sie die Daten entweder entladen und anschließend in einen Tabellenbereich laden, der mit DB2 Version 9.7 erstellt wurde, oder die Daten durch eine Onlineversetzungsoperation versetzen.

Vorgehensweise

Gehen Sie wie folgt vor, um zu ermitteln, wie viel freier Speicher unterhalb der oberen Grenze verfügbar ist:

1. Formulieren Sie eine Anweisung SELECT, die die Tabellenfunktion MON_GET_TABLESPACE enthält, um den Status Ihrer Tabellenbereiche zurückzumelden. Zum Beispiel werden durch die folgende Anweisung die Gesamtzahl der Seiten, die freien Seiten und die belegten Seiten für alle Tabellenbereiche über alle Datenbankpartitionen hinweg angezeigt:

   ```
   SELECT varchar(tbsp_name, 30) as tbsp_name,
     reclaimable_space_enabled,
     tbsp_free_pages,
     tbsp_page_top,
     tbsp_usable_pages
   FROM TABLE(MON_GET_TABLESPACE('',-2)) AS t
   ORDER BY tbsp_free_pages ASC
   ```

2. Führen Sie die Anweisung aus. Es wird eine Ausgabe wie die folgende angezeigt:

```
TBSP_NAME                      RECLAIMABLE_SPACE_ENABLED TBSP_FREE_PAGES      TBSP_PAGE_TOP        TBSP_USABLE_PAGES
------------------------------ ------------------------- -------------------- -------------------- --------------------
TEMPSPACE1                                             0                    0                    0                    1
SYSTOOLSTMPSPACE                                       0                    0                    0                    1
TBSP1                                                  1                    0                 1632                 1632
SMSDEMO                                                0                    0                    0                    1
SYSCATSPACE                                            1                 2012                10272                12284
USERSPACE1                                             1                 2496                 1696                 4064
IBMDB2SAMPLEREL                                        1                 3328                  736                 4064
TS1                                                    1                 3584                  480                 4064
TS2                                                    1                 3968                   96                 4064
TBSP2                                                  1                 3968                   96                 4064
TBSAUTO                                                1                 3968                   96                 4064
SYSTOOLSPACE                                           1                 3976                  116                 4092

  12 Satz/Sätze ausgewählt.
```

3. Anhand der folgenden Formel können Sie die Anzahl der freien Seiten unterhalb der oberen Grenze ermitteln:

   freierSpeicherUnterOB = tbsp_free_pages - (tbsp_usable_pages - tbsp_page_top)

Ergebnisse

Anhand der Informationen aus dem Bericht in Schritt 2 wäre der freie Speicher unterhalb der oberen Grenze für USERSPACE1 wie folgt zu berechnen: 2496 - (4064 - 1696) = 128 Seiten. Dies sind nur geringfügig mehr als 5 % der freien Seiten, die im Tabellenbereich insgesamt verfügbar sind.

Nächste Schritte

In diesem Fall ist ein Versuch, diesen Speicher wieder freizugeben, möglicherweise nicht der Mühe wert. Wenn Sie diese 128 Seiten dennoch freigeben wollen, könnten Sie eine Anweisung ALTER TABLESPACE USERSPACE1 REDUCE MAX ausführen. Wenn Sie diese Anweisung und anschließend die Tabellenfunktion MON_GET_TABLESPACE erneut ausführen würden, würden die folgenden Informationen angezeigt:

```
TBSP_NAME                      RECLAIMABLE_SPACE_ENABLED TBSP_FREE_PAGES      TBSP_PAGE_TOP        TBSP_USABLE_PAGES
------------------------------ ------------------------- -------------------- -------------------- --------------------
TEMPSPACE1                                             0                    0                    0                    1
```

USERSPACE1	**1**	**0**	**1568**	**1568**
SYSTOOLSTMPSPACE	0	0	0	1
TBSP1	1	0	1632	1632
SMSDEMO	0	0	0	1
SYSCATSPACE	1	2012	10272	12284
IBMDB2SAMPLEREL	1	3328	736	4064
TS1	1	3584	480	4064
TS2	1	3968	96	4064
TBSP2	1	3968	96	4064
TBSAUTO	1	3968	96	4064
SYSTOOLSPACE	1	3976	116	4092

12 Satz/Sätze ausgewählt.

Ändern von SMS-Tabellenbereichen

Für SMS-Tabellenbereiche können nach der Erstellung keine Container hinzugefügt oder die Größe von Containern geändert werden. Eine Ausnahme gilt nur, wenn Sie neue Datenpartitionen hinzufügen. In diesem Fall können Sie einem SMS-Tabellenbereich neue Container für diese Partitionen hinzufügen.

Ändern von DMS-Tabellenbereichen

Bei Verwendung von DMS-Tabellenbereichen können Sie Container hinzufügen, erweitern, neu verteilen, in der Größe ändern, löschen oder verkleinern.

Hinzufügen von DMS-Containern

Sie können einen DMS-Tabellenbereich (d. h. einen mit der Klausel MANAGED BY DATABASE erstellten Tabellenbereich) vergrößern, indem Sie dem Tabellenbereich einen oder mehrere Container hinzufügen.

Informationen zu diesem Vorgang

Wenn Sie neue Container hinzufügen und ein neues Stripe-Set erstellen, findet keine Neuverteilung statt. Ein neues Stripe-Set wird mit der Klausel BEGIN NEW STRIPE SET in der Anweisung ALTER TABLESPACE erstellt. Außerdem können Sie vorhandenen Stripe-Sets Container hinzufügen, indem Sie die Klausel ADD TO STRIPE SET in der Anweisung ALTER TABLESPACE verwenden.

Das Hinzufügen bzw. das Ändern von DMS-Containern (sowohl von Datei- als auch von Roheinheitencontainern) wird über Vorablesefunktionen parallel ausgeführt. Um die parallele Verarbeitung der Operationen zur Erstellung oder Größenänderung von Containern zu verbessern, können Sie die Anzahl der im System ausgeführten Vorablesefunktionen erhöhen. Der einzige Prozess, der nicht parallel ausgeführt werden kann, ist das Protokollieren dieser Aktionen und im Falle des Erstellens von Containern, das Kennzeichnen der Container.

Anmerkung: Um die parallele Verarbeitung der Anweisungen CREATE TABLESPACE oder ALTER TABLESPACE zu maximieren (im Hinblick auf das Hinzufügen neuer Container für einen vorhandenen Tabellenbereich), stellen Sie sicher, dass die Anzahl der Vorablesefunktionen größer oder gleich der Anzahl der hinzugefügten Container ist. Die Anzahl der Vorablesefunktionen wird über den Datenbankkonfigurationsparameter **num_ioservers** gesteuert. Die Datenbank muss gestoppt werden, damit der neue Parameterwert in Kraft treten kann. Das heißt, dass alle Anwendungen und Benutzer die Verbindung zur Datenbank trennen müssen, damit die Änderung wirksam werden kann.

Beispiel

Das folgende Beispiel zeigt, wie einem Tabellenbereich auf einem Linux- oder UNIX-Betriebssystem zwei neue Einheitencontainer (mit jeweils 10.000 Seiten) hinzugefügt werden.

```
ALTER TABLESPACE RESOURCE
  ADD (DEVICE '/dev/rhd9'  10000,
       DEVICE  '/dev/rhd10' 10000)
```

Beachten Sie, dass Sie mit der Anweisung ALTER TABLESPACE auch andere Eigenschaften des Tabellenbereichs ändern können, die sich auf die Leistung auswirken können.

Löschen von DMS-Containern

Bei einem DMS-Tabellenbereich ist es möglich, mit der Anweisung ALTER TABLESPACE einen Container aus dem Tabellenbereich zu löschen.

Informationen zu diesem Vorgang

Das Löschen eines Containers ist nur zulässig, wenn die Anzahl von EXTENTSIZE großen Speicherbereichen, die durch die Operation gelöscht werden sollen, kleiner als oder gleich der Anzahl der freien EXTENTSIZE großen Speicherbereiche oberhalb der oberen Grenze im Tabellenbereich ist. Diese Einschränkung ist notwendig, weil durch die Operation keine Seitennummern geändert werden können und daher alle Speicherbereiche bis zur oberen Grenze (einschließlich) an der gleichen logischen Position im Tabellenbereich verbleiben müssen. Das heißt, der resultierende Tabellenbereich muss ausreichend Platz haben, um alle Daten bis zur oberen Grenze einschließlich enthalten zu können. In dem Fall, dass nicht genügend freier Speicher verbleibt, empfangen Sie sofort nach Ausführung der Anweisung eine Fehlernachricht.

Wenn Container gelöscht werden, werden die verbleibenden Container neu durchnummeriert, sodass ihre Container-IDs bei 0 anfangen und sich jeweils um 1 erhöhen. Wenn alle Container in einem Stripe-Set gelöscht werden, wird das Stripe-Set aus der Zuordnung entfernt und alle nachfolgenden Stripe-Sets werden nach unten verschoben und neu nummeriert, sodass keine Lücken in den Nummern der Stripe-Sets auftreten.

Vorgehensweise

Zum Löschen eines Containers dient die Option DROP in der Anweisung ALTER TABLESPACE.

Ändern der Größe von DMS-Containern

Die Größe von Containern in einem DMS-Tabellenbereich (DMS, vom Datenbankmanager verwalteter Tabellenbereich) kann geändert werden, wenn sich der Speicherbedarf ändert. Wenn Sie die Funktion zur automatischen Größenänderung für DMS-Container verwenden, übernimmt der Datenbankmanager diese Aufgabe für Sie. Wenn Sie die Option zur automatischen Größenänderung (AUTORESIZE) nicht aktiviert haben, können Sie Anpassungen auch manuell vornehmen.

Informationen zu diesem Vorgang

Zur Vergrößerung von Containern in einem DMS-Tabellenbereich um einen bestimmten Betrag verwenden Sie die Option EXTEND des Befehls ALTER TABLESPACE. Zur Verkleinerung vorhandener Container verwenden Sie die Option RE-

DUCE. Bei der Verwendung der Option EXTEND oder REDUCE geben Sie den Betrag an, um den Sie die aktuelle Größe erhöhen oder verringern wollen. Das heißt, die Größe wird relativ zur aktuellen Größe angepasst.

Sie können auch die Option RESIZE in der Anweisung ALTER TABLESPACE verwenden. Bei Verwendung der Option RESIZE geben Sie eine neue Größe für die betroffenen Container an. Das bedeutet, dass die Größe als absolute Größe für die angegebenen Container interpretiert wird. Bei Verwendung der Option RESIZE müssen alle Container, die in der Anweisung aufgelistet werden, entweder vergrößert oder verkleinert werden. Sie können nicht innerhalb derselben Anweisung einige Container vergrößern und andere Container verkleinern.

Das Hinzufügen bzw. das Ändern von DMS-Containern (sowohl von Datei- als auch von Roheinheitencontainern) wird über Vorablesefunktionen parallel ausgeführt. Um die parallele Verarbeitung der Operationen zur Erstellung oder Größenänderung von Containern zu verbessern, können Sie die Anzahl der im System ausgeführten Vorablesefunktionen erhöhen. Der einzige Prozess, der nicht parallel ausgeführt werden kann, ist das Protokollieren dieser Aktionen und im Falle des Erstellens von Containern, das Kennzeichnen der Container.

Anmerkung: Um die parallele Verarbeitung der Anweisungen CREATE TABLESPACE oder ALTER TABLESPACE zu maximieren (im Hinblick auf das Hinzufügen neuer Container für einen vorhandenen Tabellenbereich), stellen Sie sicher, dass die Anzahl der Vorablesefunktionen größer oder gleich der Anzahl der hinzugefügten Container ist. Die Anzahl der Vorablesefunktionen wird über den Datenbankkonfigurationsparameter **num_ioservers** gesteuert. Die Datenbank muss gestoppt werden, damit der neue Parameterwert in Kraft treten kann. Das heißt, dass alle Anwendungen und Benutzer die Verbindung zur Datenbank trennen müssen, damit die Änderung wirksam werden kann.

Einschränkungen

- Jede Roheinheit kann nur als ein einziger Container verwendet werden.
- Die Roheinheit ist nach ihrer Erstellung festgelegt.
- Wenn Sie beabsichtigen, einen Container für eine Roheinheit mithilfe der Option RESIZE bzw. EXTEND zu vergrößern, müssen Sie die Größe der Roheinheit zunächst überprüfen, um sicherzugehen, dass Sie nicht versuchen, den Einheitencontainer über die Größe der Roheinheit hinaus zu vergrößern.
- In DMS-Tabellenbereichen muss die Länge eines Containers mindestens das Doppelte der Länge der Seiten des Speicherbereichs betragen. Der maximale Wert für die Größe eines Containers hängt vom Betriebssystem ab.

Beispiel

Beispiel 1: Vergrößern von Dateicontainern. Das folgende Beispiel zeigt, wie Dateicontainer (die jeweils in einer Größe von 1.000 Seiten vorliegen) in einem Tabellenbereich auf einem Windows-System vergrößert werden:

```
ALTER TABLESPACE PERSNEL
   EXTEND (FILE 'e:\wrkhist1'  200
           FILE  'f:\wrkhist2'  200)
```

Bei beiden Dateien vergrößert sich der Umfang von 1.000 Seiten auf 1.200 Seiten. Der Inhalt des Tabellenbereichs kann in den Containern neu verteilt werden. Während der Neuverteilung ist der Zugriff auf den Tabellenbereich nicht eingeschränkt.

Beispiel 2: Vergrößern von Einheitencontainern. Das folgende Beispiel zeigt, wie zwei Einheitencontainer (die jeweils mit einer Größe von 1.000 Seiten vorliegen) in einem Tabellenbereich unter einem Linux- oder UNIX-Betriebssystem vergrößert werden:

```
ALTER TABLESPACE HISTORY
  RESIZE (DEVICE '/dev/rhd7'  2000,
          DEVICE  '/dev/rhd8' 2000)
```

Bei beiden Einheiten vergrößert sich der Umfang von 1.000 Seiten auf 2.000 Seiten. Der Inhalt des Tabellenbereichs kann in den Containern neu verteilt werden. Während der Neuverteilung ist der Zugriff auf den Tabellenbereich nicht eingeschränkt.

Beispiel 3: Verkleinern mithilfe der Option REDUCE. Das folgende Beispiel zeigt, wie ein Dateicontainer (der mit einer Größe von 1.000 Seiten vorliegt) in einem Tabellenbereich unter einem Windows-Betriebssystem verkleinert wird:

```
ALTER TABLESPACE PAYROLL
  REDUCE (FILE 'd:\hldr\finance'  200)
```

Durch diese Aktion wird die Datei von 1.000 Seiten auf die Größe von 800 Seiten verkleinert.

Neuverteilung von DMS-Containern

Der Prozess der Neuverteilung besteht im Versetzen von EXTENTSIZE großen Speicherbereichen des Tabellenbereichs von einer Position an eine andere. Dies geschieht, um die einheitenübergreifende Speicherung (Striping) der Daten innerhalb des Tabellenbereichs beizubehalten. In der Regel führen Sie eine Neuverteilung der Daten in einem Tabellenbereich aus, wenn Sie in einer Datenbank Speicherpfade hinzufügen oder löschen.

Auswirkung des Hinzufügens oder Löschens von Containern auf die Neuverteilung

Bei der Erstellung eines Tabellenbereichs wird die zugehörige Tabellenbereichszuordnung erstellt und alle Anfangscontainer werden so ausgerichtet, dass sie in Stripe 0 beginnen (Stripe - einheitenübergreifend gespeicherter Datenblock). Dies bedeutet, dass die Daten gleichmäßig über sämtliche Tabellenbereichscontainer verteilt gespeichert werden, bis die einzelnen Container gefüllt sind. (Siehe Beispiel 1 („Vorher").)

Durch das Hinzufügen eines Containers, der kleiner als vorhandene Container ist, wird eine ungleichmäßige Verteilung der Daten verursacht. Dies kann dazu führen, dass parallele E/A-Operationen wie das Vorablesen von Daten weniger effizient ausgeführt werden, als sie es bei Containern gleicher Größe könnten.

Wenn einem Tabellenbereich neue Container hinzugefügt oder vorhandene Container erweitert werden, erfolgt eine Neuverteilung der Tabellenbereichsdaten, wenn der neue Bereich unterhalb der *oberen Grenze* für den Tabellenbereich hinzugefügt wird. Wenn neuer Speicherbereich oberhalb der oberen Grenze hinzugefügt wird oder wenn Sie ein neues Stripe-Set erstellen, erfolgt keine automatische Neuverteilung. Eine Neuverteilung, die erfolgt, um hinzugefügten Speicher zu nutzen, wird als *progressive Neuverteilung* bezeichnet. Dabei beginnt die Speicherbereichsverschiebung mit dem Speicherbereich (Extent) 0, d. h. dem ersten Speicherbereich im Tabellenbereich, und schreitet voran bis zu dem Speicherbereich unmittelbar unterhalb der oberen Grenze.

Durch Hinzufügen eines Containers wird beinahe immer Speicherplatz unterhalb der oberen Grenze hinzugefügt. Daher ist häufig eine Neuverteilung erforderlich, wenn Sie einen Container hinzufügen. Sie können veranlassen, dass neue Container oberhalb der oberen Grenze hinzugefügt werden. Dies bietet die Möglichkeit, keine Neuverteilung des Inhalts des Tabellenbereichs durchzuführen. Ein Vorteil dieser Methode ist, dass der neue Container sofort zur Verwendung verfügbar ist. Das Hinzufügen von Containern zu einem Tabellenbereich ohne Neuverteilung geschieht durch Hinzufügen eines neuen *Stripe-Sets*. Ein Stripe-Set ist eine Gruppe von Containern in einem Tabellenbereich, die Datenblöcke enthält, die einheitenübergreifend in ihr gespeichert sind, und die von den anderen Containern getrennt ist, die zu diesem Tabellenbereich gehören. Die vorhanden Container in den vorhandenen Stripe-Sets bleiben unberührt, und die Container, die Sie hinzufügen, werden Teil eines neuen Stripe-Set. Verwenden Sie die Klausel BEGIN NEW STRIPE SET in der Anweisung ALTER TABLESPACE, wenn Sie Container ohne Neuverteilung hinzufügen wollen.

Wenn Container aus einem Tabellenbereich gelöscht werden, erfolgt automatisch eine Neuverteilung, wenn sich in dem Bereich, der gelöscht wird, Daten befinden. In diesem Fall wird die Neuverteilung als *regressive Neuverteilung* bezeichnet. Dabei beginnt die Speicherbereichsverschiebung an der oberen Grenze und schreitet abwärts bis zum ersten Speicherbereich im Tabellenbereich voran.

Vor dem Start der Neuverteilung wird eine neue Tabellenbereichszuordnung auf der Grundlage der vorgenommenen Containeränderungen erstellt. Die Neuverteilungsfunktion versetzt EXTENTSIZE große Speicherbereiche von ihrer Position, die durch die aktuelle Zuordnung festgelegt ist, an die Position, die durch die neue Zuordnung festgelegt wird.

Progressive Neuverteilung

Die Neuverteilungsfunktion beginnt mit dem Speicherbereich 0 und versetzt jeweils einen Speicherbereich gleichzeitig, bis der Speicherbereich, der die obere Grenze enthält, versetzt wurde. Beim Versetzen der einzelnen Speicherbereiche wird die aktuelle Zuordnung stückweise in das Aussehen der neuen Zuordnung geändert. Wenn die Neuverteilung abgeschlossen ist, sollten die aktuelle Zuordnung und die neue Zuordnung bis zu dem Stripe identisch aussehen, der die obere Grenze enthält. Die aktuelle Zuordnung wird dann vollständig an das Aussehen der neuen Zuordnung angeglichen und der Neuverteilungsprozess ist abgeschlossen. Wenn die Position eines Speicherbereichs in der aktuellen Zuordnung mit seiner Position in der neuen Zuordnung übereinstimmt, wird der Speicherbereich nicht versetzt, und es finden keine E/A-Operationen statt.

Wenn ein neuer Container hinzugefügt wird, hängt die Positionierung dieses Containers innerhalb der neuen Zuordnung von seiner Größe sowie der Größe der anderen Container in seinem Stripe-Set ab. Wenn der Container groß genug ist, um im ersten Stripe des Stripe-Sets zu beginnen und im letzten Stripe (oder dahinter) des Stripe-Sets zu enden, wird er in dieser Weise angeordnet (siehe Beispiel 1 („Nachher")). Wenn der Container dazu nicht groß genug ist, wird er in der Zuordnung so positioniert, dass er im letzten Stripe des Stripe-Set endet (siehe Beispiel 3). Dies geschieht, um das Volumen der Daten zu minimieren, die neu verteilt werden müssen.

Während dieser Neuverteilung der Daten wird der Zugriff auf den Tabellenbereich nicht eingeschränkt. Objekte können wie gewöhnlich gelöscht, erstellt, mit Daten gefüllt und abgefragt werden. Allerdings kann die Operation der Neuverteilung erhebliche Auswirkungen auf die Leistung haben. Wenn mehr als ein Container hin-

zugefügt werden muss und Sie planen, die Container neu zu verteilen, sollten diese Container gleichzeitig innerhalb einer einzigen Anweisung ALTER TABLESPACE hinzugefügt werden, um zu vermeiden, dass der Datenbankmanager die Daten mehr als einmal neu verteilen muss.

Anmerkung: In den folgenden Beispielen wird bei den Containergrößen die Größe der Containerkennung (container tag) nicht berücksichtigt. Die Containergrößen sind sehr klein und dienen lediglich zu Veranschaulichungszwecken. Sie stellen keine empfohlenen Containergrößen dar. Die Beispiele zeigen Container unterschiedlicher Größen innerhalb eines Tabellenbereichs, jedoch wird empfohlen, Container gleicher Größe zu verwenden.

Regressive Neuverteilung

Die Neuverteilungsfunktion beginnt mit dem Speicherbereich, der die obere Grenze enthält, und versetzt jeweils einen Speicherbereich gleichzeitig, bis Speicherbereich 0 versetzt wurde. Beim Versetzen der einzelnen Speicherbereiche wird die aktuelle Zuordnung stückweise in das Aussehen der neuen Zuordnung geändert. Wenn die Position eines Speicherbereichs in der aktuellen Zuordnung mit seiner Position in der neuen Zuordnung übereinstimmt, wird der Speicherbereich nicht versetzt und es finden keine E/A-Operationen statt.

Beispiele

Beispiel 1 (vorher): Tabellenbereichslayout vor dem Hinzufügen von Containern

Wenn Sie einen Tabellenbereich mit drei Containern und einem EXTENTSIZE-Wert von 10 erstellen und die Container 60, 40 bzw. 80 Seiten (entsprechend EXTENTSIZE-Werten von 6, 4 und 8) groß sind, wird der Tabellenbereich mit einer Zuordnung erstellt, die sich wie in Abb. 19 auf Seite 237 darstellen lässt.

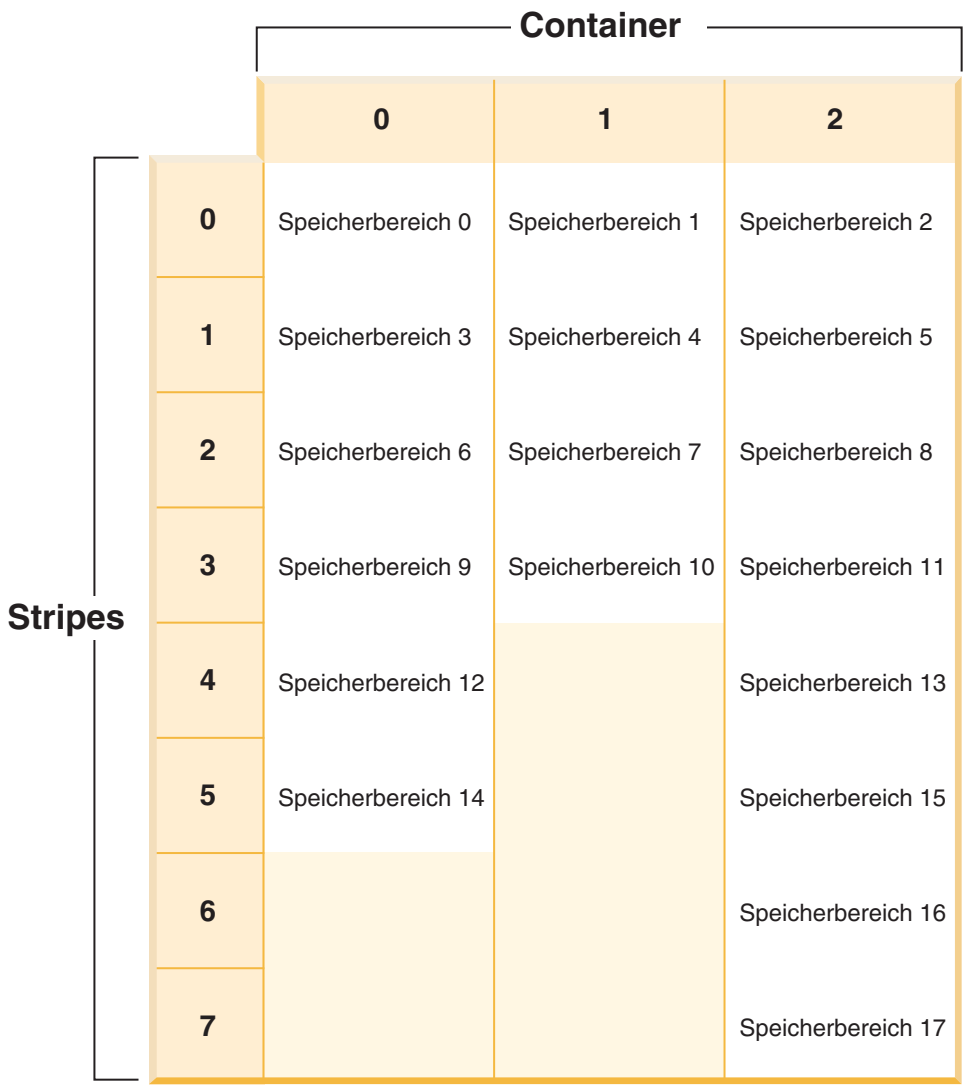


Abbildung 19. Tabellenbereich mit drei Containern und 18 Speicherbereichen

Die entsprechende Tabellenbereichszuordnung, wie sie in einer Momentaufnahme des Tabellenbereichs gezeigt würde, sieht folgendermaßen aus:

```
Range     Stripe  Stripe  Max        Max     Start    End    Adj.   Containers
Number    Set     Offset  Extent     Page    Stripe   Stripe
   [0]       [0]      0      11      119        0        3    0   3 (0, 1, 2)
   [1]       [0]      0      15      159        4        5    0   2 (0, 2)
   [2]       [0]      0      17      179        6        7    0   1 (2)
```

Die Spaltenüberschriften in der Tabellenbereichszuordnung heißen 'Range Number' (Bereichsnummer), 'Stripe Set', 'Stripe Offset', 'Maximum extent number addressed by the range' (höchste Speicherbereichsnummer, die durch den Bereich adressiert wird), 'Maximum page number addressed by the range' (höchste Seitennummer, die durch den Bereich adressiert wird), 'Start Stripe' (Anfangsstripe), 'End Stripe' (Endstripe), 'Range adjustment' (Bereichsanpassung) und 'Container list' (Containerliste).

Beispiel 1 (nachher): Hinzufügen eines Containers mit der Folge einer progressiven Neuverteilung

Wenn dem Tabellenbereich in Beispiel 1 ein 80 Seiten großer Container hinzugefügt wird, ist der Container groß genug, um im ersten Stripe (Stripe 0) zu beginnen und im letzten Stripe (Stripe 7) zu enden. Er wird so positioniert, dass er im ersten Stripe beginnt. Der resultierende Tabellenbereich lässt sich wie in Abb. 20 darstellen.

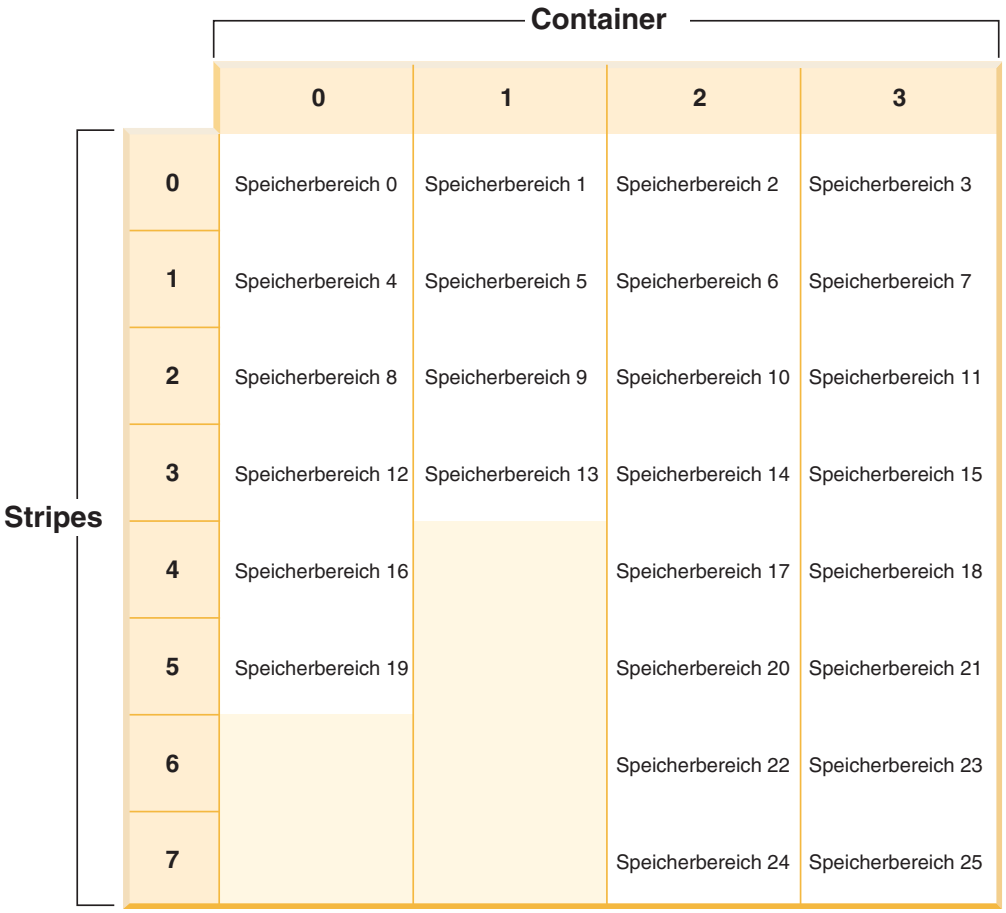


Abbildung 20. Tabellenbereich mit vier Containern und 26 Speicherbereichen

Die entsprechende Tabellenbereichszuordnung, wie sie in einer Momentaufnahme des Tabellenbereichs gezeigt würde, sieht folgendermaßen aus:

```
Range     Stripe  Stripe  Max        Max    Start    End    Adj.   Containers
Number    Set     Offset  Extent     Page   Stripe   Stripe
   [0]       [0]       0      15      159       0       3    0   4 (0, 1, 2, 3)
   [1]       [0]       0      21      219       4       5    0   3 (0, 2, 3)
   [2]       [0]       0      25      259       6       7    0   2 (2, 3)
```

Wenn sich die obere Grenze im Speicherbereich 14 befindet, startet die Neuverteilungsfunktion bei Speicherbereich 0 und versetzt alle Speicherbereiche bis einschließlich Speicherbereich 14. Die Position von Speicherbereich 0 innerhalb der beiden Zuordnungen ist identisch, sodass dieser Speicherbereich nicht versetzt werden muss. Das Gleiche gilt für die Speicherbereiche 1 und 2. Speicherbereich 3 muss versetzt werden, sodass der Speicherbereich von der alten Position (zweiter Speicherbereich in Container 0) gelesen und an die neue Position (erster Speicherbereich in Container 3) geschrieben wird. Jeder Speicherbereich nach diesem bis einschließlich Speicherbereich 14 wird versetzt. Wenn Speicherbereich 14 versetzt wurde, sieht die aktuelle Zuordnung wie die neue Zuordnung aus und die Neuverteilungsfunktion wird beendet.

Wenn die Zuordnung so geändert wird, dass sämtlicher neu hinzugefügter Speicherplatz über der oberen Grenze liegt, ist keine Neuverteilung erforderlich und der gesamte Speicherplatz ist sofort zur Verwendung verfügbar. Wenn die Zuordnung so geändert wird, dass einiger Speicherplatz oberhalb der oberen Grenze liegt, ist der Speicherplatz in den Stripes oberhalb der oberen Grenze verfügbar. Der übrige Speicherplatz ist erst verfügbar, wenn die Neuverteilung abgeschlossen ist.

Wenn Sie einen Container erweitern, arbeitet die Neuverteilung ähnlich. Wenn ein Container so erweitert wird, dass er über den letzten Stripe in seinem Stripe-Set hinausreicht, wird das Stripe-Set entsprechend vergrößert und die folgenden Stripe-Sets werden entsprechend nach hinten verschoben. Infolgedessen reicht der Container nicht in eines der nachfolgenden Stripe-Sets hinein.

Beispiel 2: Erweitern eines Containers

Betrachten Sie den Tabellenbereich aus Beispiel 1. Wenn Sie den Container 1 von 40 Seiten auf 80 Seiten erweitern, sieht der neue Tabellenbereich wie in Abb. 21 auf Seite 240 aus.

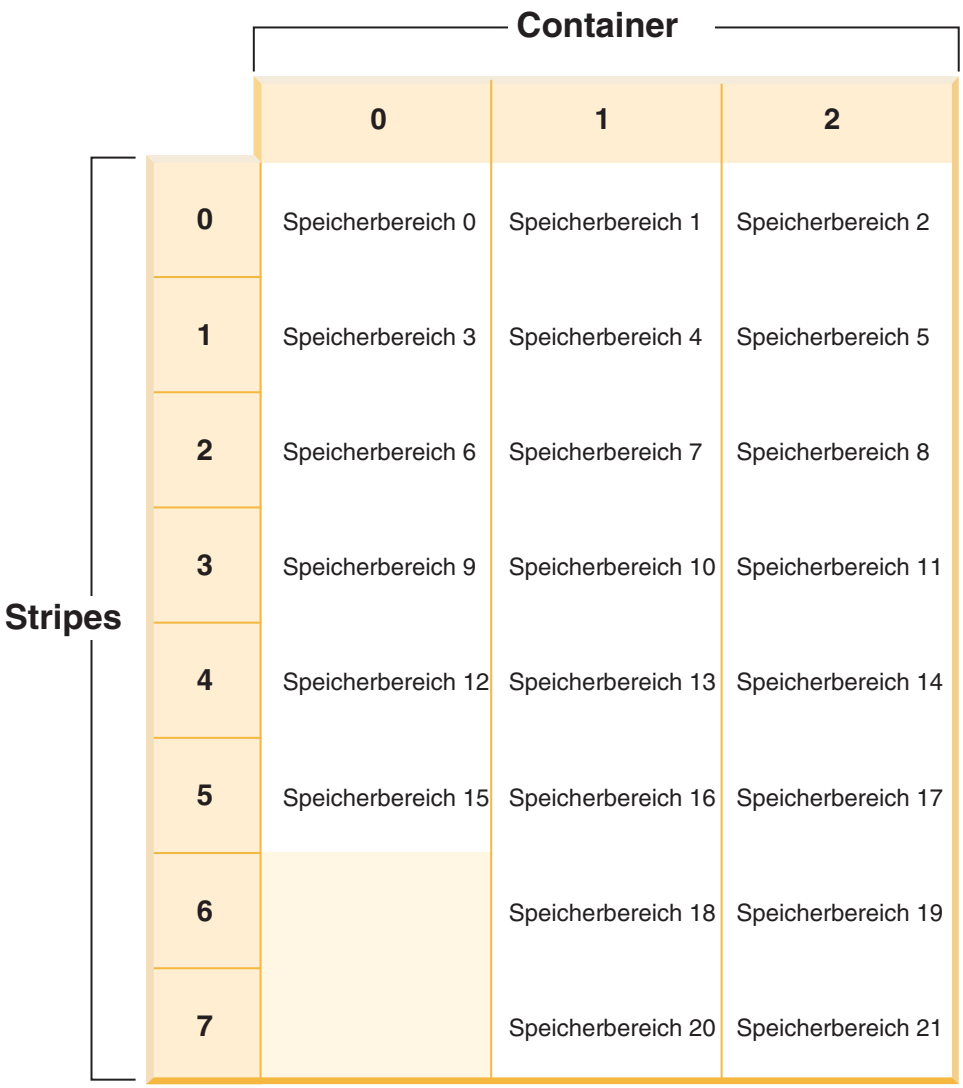


Abbildung 21. Tabellenbereich mit drei Containern und 22 Speicherbereichen

Die entsprechende Tabellenbereichszuordnung, wie sie in einer Momentaufnahme des Tabellenbereichs gezeigt würde, sieht folgendermaßen aus:

```
Range     Stripe  Stripe  Max        Max    Start    End    Adj.   Containers
Number    Set     Offset  Extent     Page   Stripe   Stripe
   [0]       [0]      0      17      179         0        5    0   3 (0, 1, 2)
   [1]       [0]      0      21      219         6        7    0   2 (1, 2)
```

Beispiel 3: Hinzufügen eines Containers, der nicht groß genug ist, um im ersten Stripe zu beginnen und im letzten Stripe zu enden

Betrachten Sie den Tabellenbereich aus Beispiel 1. Wenn ein 50 Seiten großer Container (fünf EXTENTSIZE-Größen) hinzugefügt wird, wird der Container der neuen Zuordnung wie folgt hinzugefügt. Der Container ist nicht groß genug, um im ersten Stripe (Stripe 0) zu beginnen und im letzten Stripe (Stripe 7) oder dahinter zu enden. Daher wird er in der Weise angeordnet, dass er im letzten Stripe endet. (Siehe Abb. 22 auf Seite 241.)

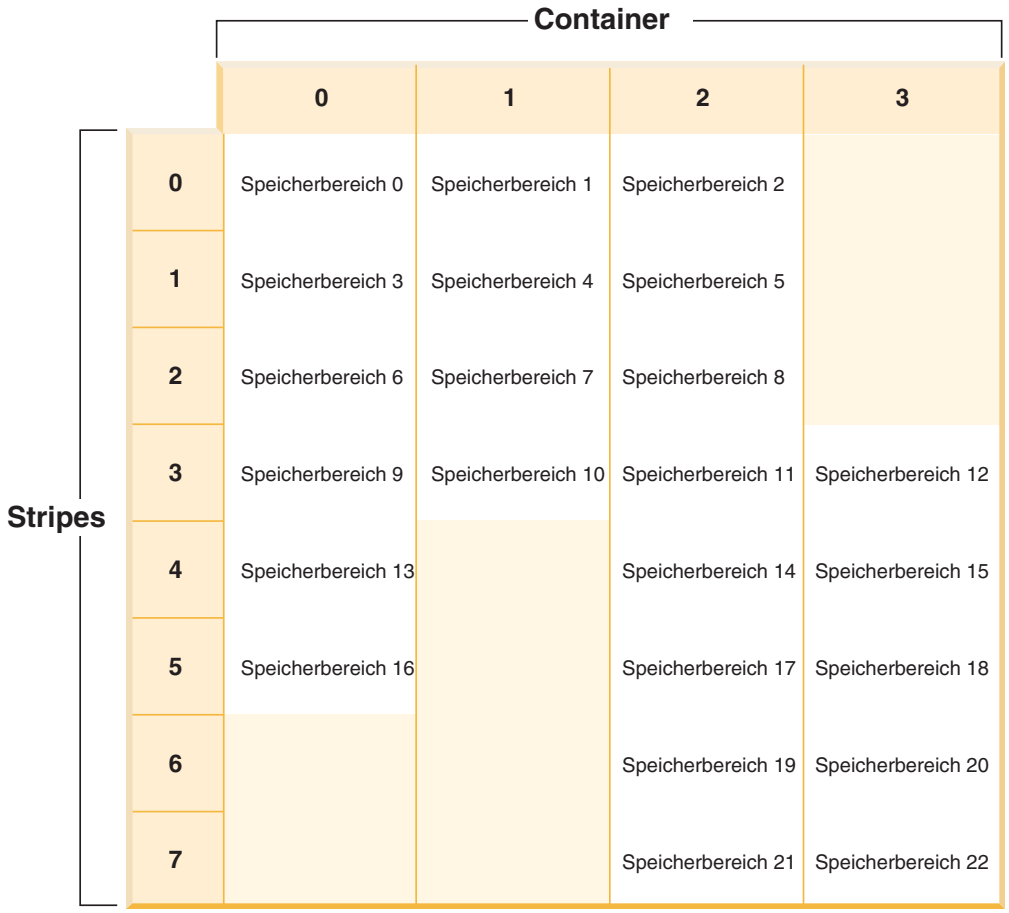


Abbildung 22. Tabellenbereich mit vier Containern und 23 Speicherbereichen

Die entsprechende Tabellenbereichszuordnung, wie sie in einer Momentaufnahme des Tabellenbereichs gezeigt würde, sieht folgendermaßen aus:

```
Range     Stripe  Stripe  Max       Max    Start    End    Adj.   Containers
Number    Set     Offset  Extent    Page   Stripe   Stripe
   [0]      [0]      0       8       89       0        2    0   3 (0, 1, 2)
   [1]      [0]      0      12      129       3        3    0   4 (0, 1, 2, 3)
   [2]      [0]      0      18      189       4        5    0   3 (0, 2, 3)
   [3]      [0]      0      22      229       6        7    0   2 (2, 3)
```

Verwenden Sie zur Erweiterung eines Containers die Klausel EXTEND oder RESIZE in der Anweisung ALTER TABLESPACE. Zum Hinzufügen eines Containers und Neuverteilen der Daten dient die Klausel ADD in der Anweisung ALTER TABLESPACE. Wenn Sie einem Tabellenbereich, der bereits mehr als ein Stripe-Set hat, einen Container hinzufügen, können Sie angeben, welchem Stripe-Set der Container hinzugefügt werden soll. Dazu verwenden Sie die Klausel ADD TO STRIPE SET in der Anweisung ALTER TABLESPACE. Wenn Sie kein Stripe-Set angeben, wird der Container standardmäßig dem aktuellen Stripe-Set hinzugefügt. Das aktuelle Stripe-Set ist das zuletzt erstellte Stripe-Set, nicht das, dem zuletzt Speicherplatz hinzugefügt wurde.

Jede Änderung an einem Stripe-Set kann eine Neuverteilung in diesem Stripe-Set und den anderen nachfolgenden Stripe-Sets zur Folge haben.

Sie können den Fortschritt einer Neuverteilung mithilfe von Momentaufnahmen der Tabellenbereiche überwachen. Eine Momentaufnahme eines Tabellenbereichs

kann Informationen über eine Neuverteilung liefern, wie zum Beispiel den Startzeitpunkt der Neuverteilung, die Anzahl der versetzten Speicherbereiche und die Anzahl der zu versetzenden Speicherbereiche.

Beispiel 4: Löschen eines Containers mit der Folge einer regressiven Neuverteilung

Anmerkung: In den folgenden Beispielen wird bei den Containergrößen die Größe der Containerkennung (container tag) nicht berücksichtigt. Die Containergrößen sind sehr klein und dienen lediglich zu Veranschaulichungszwecken. Sie stellen keine empfohlenen Containergrößen dar. Die Beispiele zeigen Container unterschiedlicher Größen innerhalb eines Tabellenbereichs, jedoch dient dies nur der Veranschaulichung. Zu empfehlen ist die Verwendung von Containern gleicher Größe.

Betrachten Sie zum Beispiel einen Tabellenbereich mit drei Containern und einem EXTENTSIZE-Wert 10. Die Container sind 20, 50 und 50 Seiten (d. h. 2, 5 und 5 EXTENTSIZE-Größen groß). Der Tabellenbereich lässt sich wie in Abb. 23 darstellen.

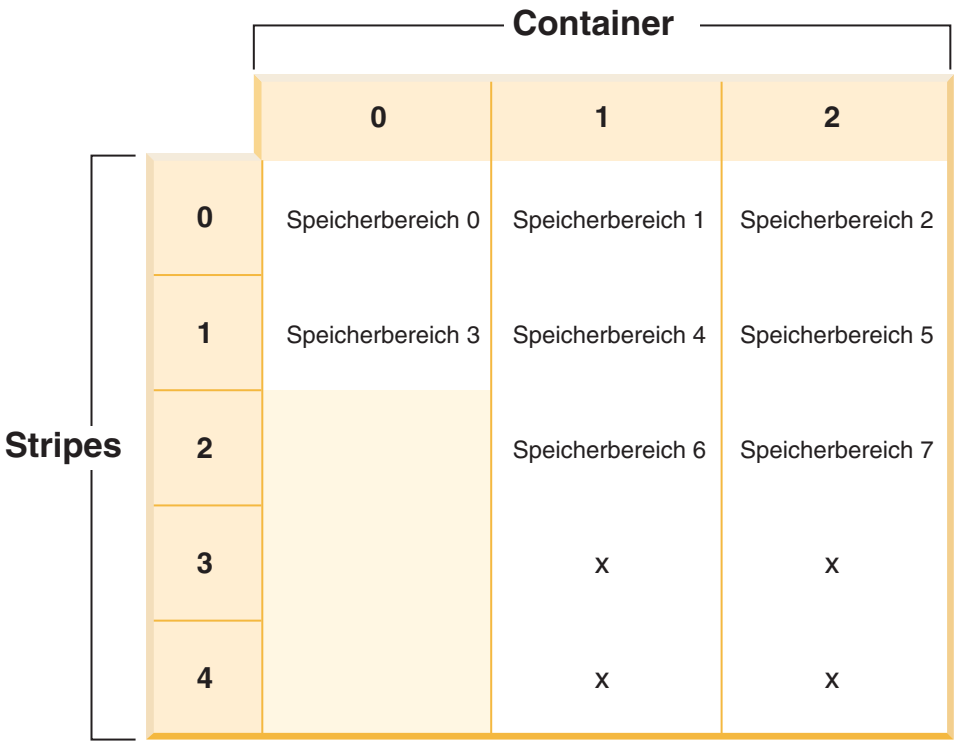


Abbildung 23. Tabellenbereich mit zwölf Speicherbereichen und vier Speicherbereichen ohne Daten

Ein X zeigt an, dass an der Stelle ein Speicherbereich vorhanden ist, der jedoch keine Daten enthält.

Wenn Sie den Container 0 löschen, der zwei Speicherbereiche enthält, müssen über der oberen Grenze mindestens zwei freie Speicherbereiche vorhanden sein. Die obere Grenze ist der Speicherbereich 7, sodass vier freie Speicherbereiche verbleiben. In diesem Fall können Sie also den Container 0 löschen.

Die entsprechende Tabellenbereichszuordnung, wie sie in einer Momentaufnahme des Tabellenbereichs gezeigt würde, sieht folgendermaßen aus:

```
Range     Stripe  Stripe  Max        Max    Start    End    Adj.   Containers
Number    Set     Offset  Extent     Page   Stripe   Stripe
   [0]       [0]       0       5       59        0        1    0   3 (0, 1, 2)
   [1]       [0]       0      11      119        2        4    0   2 (1, 2)
```

Nach dem Löschen enthält der Tabellenbereich nur Container 0 und Container 1. Der neue Tabellenbereich lässt sich wie in Abb. 24 darstellen.

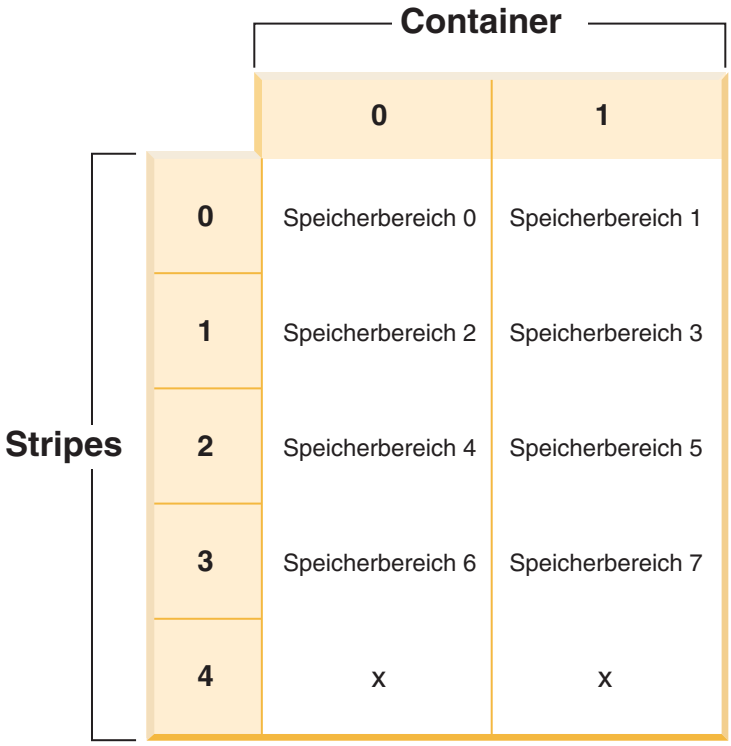


Abbildung 24. Tabellenbereich nach dem Löschen eines Containers

Die entsprechende Tabellenbereichszuordnung, wie sie in einer Momentaufnahme des Tabellenbereichs gezeigt würde, sieht folgendermaßen aus:

```
Range     Stripe  Stripe  Max        Max    Start    End    Adj.   Containers
Number    Set     Offset  Extent     Page   Stripe   Stripe
   [0]       [0]       0       9       99        0        4    0   2 (0, 1)
```

Beispiel 5: Hinzufügen eines neuen Stripe-Sets

Wenn Sie einen Tabellenbereich mit drei Containern und einem EXTENTSIZE-Wert von 10 haben, und die Container 30, 40 und 40 Seiten (3, 4 und 4 EXTENTSIZE-Werte) groß sind, kann der Tabellenbereich wie in Abb. 25 auf Seite 244 dargestellt werden.

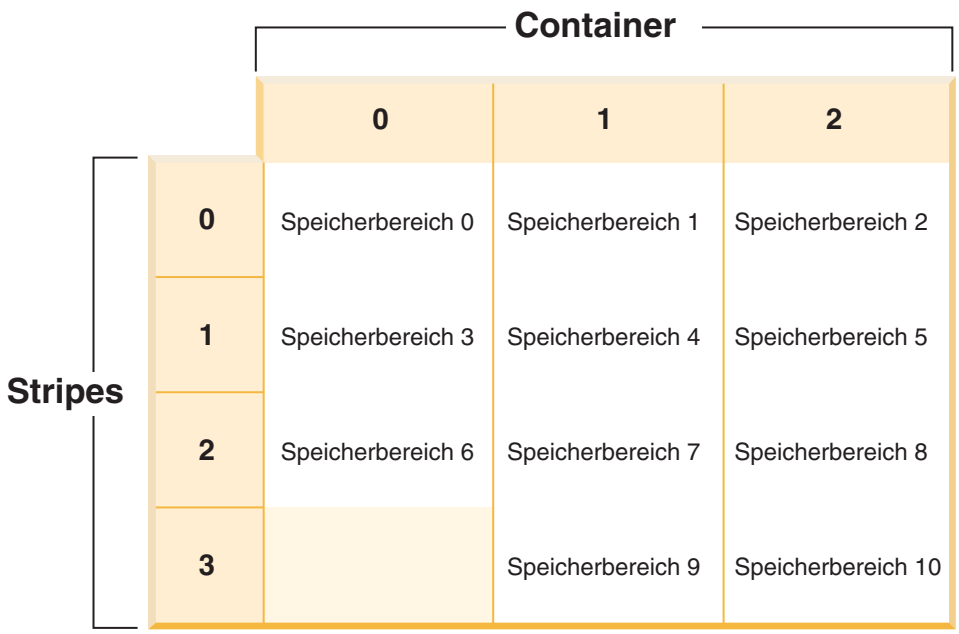


Abbildung 25. Tabellenbereich mit drei Containern und 11 Speicherbereichen

Die entsprechende Tabellenbereichszuordnung, wie sie in einer Momentaufnahme des Tabellenbereichs gezeigt würde, sieht folgendermaßen aus:

```
Range     Stripe  Stripe  Max       Max    Start    End    Adj.   Containers
Number    Set     Offset  Extent    Page   Stripe   Stripe
   [0]       [0]      0       8      89       0        2    0   3 (0, 1, 2)
   [1]       [0]      0      10     109       3        3    0   2 (1, 2)
```

Wenn Sie zwei neue Container mit der Klausel BEGIN NEW STRIPE SET hinzufügen, die 30 und 40 Seiten (3 und 4 EXTENTSIZE-Größen) groß sind, werden die vorhandenen Bereiche (Ranges) davon nicht berührt. Stattdessen wird eine neue Gruppe von Bereichen (Ranges) erstellt. Diese neue Gruppe von Bereichen ist ein Stripe-Set und das zuletzt erstellte Stripe-Set wird als aktuelles Stripe-Set bezeichnet. Nach dem Hinzufügen der beiden neuen Container sieht der Tabellenbereich wie in Abb. 26 auf Seite 245 aus.

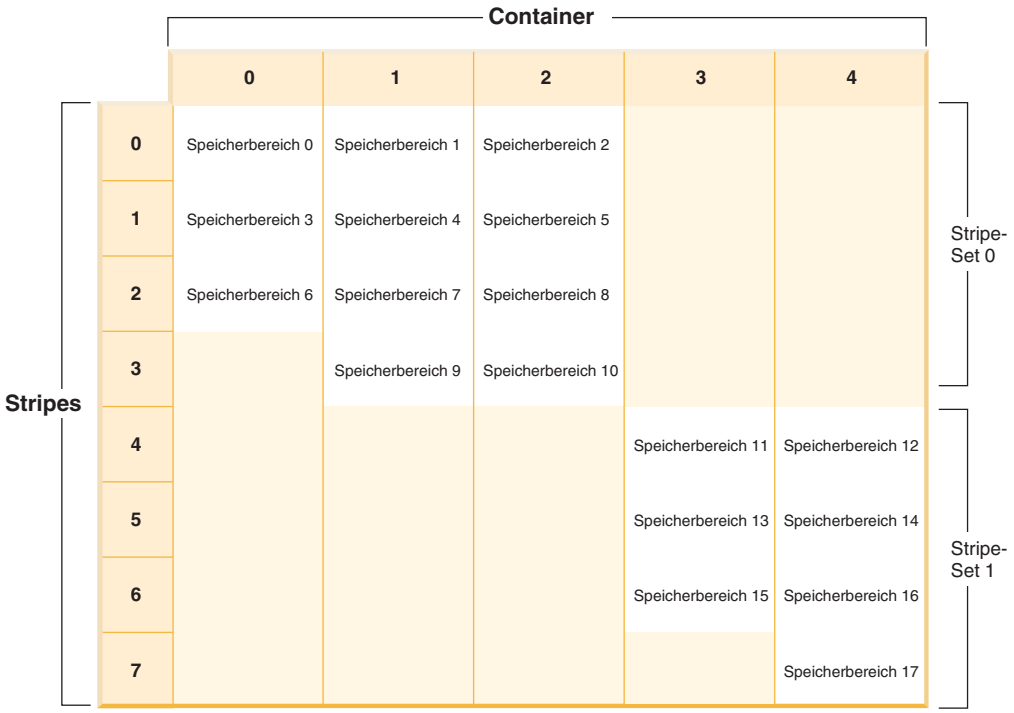


Abbildung 26. Tabellenbereich mit zwei Stripe-Sets

Die entsprechende Tabellenbereichszuordnung, wie sie in einer Momentaufnahme des Tabellenbereichs gezeigt würde, sieht folgendermaßen aus:

```
Range     Stripe  Stripe  Max       Max    Start    End    Adj.   Containers
Number    Set     Offset  Extent    Page   Stripe   Stripe
   [0]      [0]      0       8       89       0       2    0   3 (0, 1, 2)
   [1]      [0]      0      10      109       3       3    0   2 (1, 2)
   [2]      [1]      4      16      169       4       6    0   2 (3, 4)
   [3]      [1]      4      17      179       7       7    0   1 (4)
```

Wenn Sie einem Tabellenbereich neue Container hinzufügen und Sie die Klausel TO STRIPE SET mit der Klausel ADD *nicht* verwenden, werden die Container dem aktuellen Stripe-Set (d. h. dem Stripe-Set mit der höchsten Nummer) hinzugefügt. Mit der Klausel ADD TO STRIPE SET können Sie jedem Stripe-Set im Tabellenbereich Container hinzufügen. Sie müssen ein gültiges Stripe-Set angeben.

Der Datenbankmanager verwaltet die Stripe-Sets mithilfe der Tabellenbereichszuordnung. Durch das Hinzufügen von Containern ohne Neuverteilung wächst die Zuordnung in der Regel schneller als bei Durchführung einer Neuverteilung der Container. Wenn die Tabellenbereichszuordnung zu groß wird, empfangen Sie den Fehler SQL0259N, wenn Sie versuchen, weitere Container hinzuzufügen.

Überwachen einer Ausgleichsoperation für einen Tabellenbereich

Mit der Tabellenfunktion **MON_GET_REBALANCE_STATUS** kann der Fortschritt von Ausgleichsoperationen für eine Datenbank überwacht werden.

Informationen zu diesem Vorgang

Diese Prozedur gibt nur Daten für einen Tabellenbereich zurück, wenn eine Ausgleichsoperation in Bearbeitung ist. Andernfalls werden keine Daten zurückgegeben.

Vorgehensweise

Gehen Sie wie folgt vor, um eine Ausgleichsoperation für einen Tabellenbereich zu überwachen:

Setzen Sie die Tabellenfunktion **MON_GET_REBALANCE_STATUS** mit den Parametern **tbsp_name** und **dbpartitionnum** ab:

```
select
varchar(tbsp_name, 30) as tbsp_name,
   dbpartitionnum,
   member,
   rebalancer_mode,
   rebalancer_status,
   rebalancer_extents_remaining,
   rebalancer_extents_processed,
   rebalancer_start_timefrom table(mon_get_rebalance_status(NULL,-2)) as t
```

Ergebnisse

Dies ist eine für die Überwachung des Fortschritts einer Ausgleichsoperation für einen Tabellenbereich typische Ausgabe:

```
TBSP_NAME                    DBPARTITIONNUM MEMBER REBALANCER_MODE
---------------------------- -------------- ------ ----------------------------
SYSCATSPACE                               0      0 REV_REBAL


REBALANCER_STATUS REBALANCER_EXTENTS_REMAINING REBALANCER_EXTENTS_PROCESSED REBALANCER_START_TIME
----------------- --------------------------- --------------------------- --------------------------
ACTIVE                                   6517                           4 2011-12-01-12.08.16.000000

  1 Satz/Sätze ausgewählt.
```

Freigeben von nicht belegtem Speicher in DMS-Tabellenbereichen

Sie können nicht belegten Speicher in einem DMS-Tabellenbereich freigeben, indem Sie den Datenbankmanager anweisen, die belegten Speicherbereiche weiter unten im Tabellenbereich zu konsolidieren. Dies hat zudem den Effekt, dass die obere Grenze gesenkt wird. Zur Verkleinerung von Containern in einem DMS-Tabellenbereich muss darüber hinaus eine separate REDUCE-Operation ausgeführt werden.

Vorbereitende Schritte

Der Tabellenbereich muss ein mit DB2 Version 9.7 oder einer späteren Version erstellter DMS-Tabellenbereich sein. Die Funktion des konsolidierbaren Speichers ist nicht in Tabellenbereichen verfügbar, die mit früheren Versionen des DB2-Produkts erstellt wurden. Mithilfe der Tabellenfunktion MON_GET_TABLESPACE lässt sich prüfen, welche Tabellenbereiche in einer Datenbank die Funktion des konsolidierbaren Speichers unterstützen.

Informationen zu diesem Vorgang

Wenn Sie nicht belegten Speicher in einem DMS-Tabellenbereich freigeben wollen, müssen Sie zunächst eine Operation einleiten, die bewirkt, dass (EXTENTSIZE große) Speicherbereiche in der Tabelle neu angeordnet werden, sodass die freien Speicherbereiche weiter unten im Tabellenbereich genutzt werden. Dies geschieht mithilfe der Klausel LOWER HIGH WATER MARK der Anweisung ALTER TABLESPACE. Als Nächstes können Sie die Größe der Container in dem Tabellenbereich um einen angegebenen Betrag verringern.

Wenn Sie die Größe von Containern in einem DMS-Tabellenbereich verringern, müssen Sie die Namen der zu verkleinernden Container angeben oder die Klausel ALL CONTAINERS verwenden.

Einschränkungen

- Sie können Speicher nur in Tabellenbereichen freigeben, die mit DB2 Version 9.7 oder späteren Versionen erstellt wurden.
- Wenn Sie die Klausel REDUCE oder die Klausel LOWER HIGH WATER MARK in der Anweisung ALTER TABLESPACE angeben, können Sie keine anderen Parameter angeben.
- Wenn sich der Speicherbereich, der die Seite enthält, die zurzeit als obere Grenze angegeben ist, im Status „Löschen anstehend" befindet, kann der Versuch, die obere Grenze durch Versetzen von Speicherbereichen zu senken, fehlschlagen. In diesem Fall wird die Nachricht ADM6008I protokolliert. Speicherbereiche im Status „Löschen anstehend" können aus Wiederherstellbarkeitsgründen nicht immer versetzt werden. Solche Speicherbereiche werden schließlich durch normale Datenbankwartungsprozesse freigegeben, sodass sie dann versetzt werden können.

Vorgehensweise

1. Verwenden Sie die Anweisung ALTER TABLESPACE mit der Klausel LOWER HIGH WATER MARK, um die obere Grenze so weit wie möglich durch eine Neuanordnung von Speicherbereichen innerhalb des Tabellenbereichscontainers zu senken.
2. Verwenden Sie die Anweisung ALTER TABLESPACE mit der Klausel REDUCE, um die Größe einiger oder aller Container um einen angegebenen Betrag zu verringern.

Beispiel

Beispiel 1: Senken der oberen Grenze und Verkleinern aller Container um 5 MB. Im folgenden Beispiel wird die obere Grenze für Tabellenbereich `ts` gesenkt und die Größe aller Container im Tabellenbereich um 5 MB verringert.

```
ALTER TABLESPACE ts LOWER HIGH WATER MARK
ALTER TABLESPACE ts REDUCE (ALL CONTAINERS 5 M)
```

Beispiel 2: Senken der oberen Grenze und Verkleinern des Containers „Container1" um 2000 Seiten. Im folgenden Beispiel wird die obere Grenze für Tabellenbereich `ts` gesenkt und die Größe von „Container1" um 2000 Seiten verringert.

```
ALTER TABLESPACE ts LOWER HIGH WATER MARK
ALTER TABLESPACE ts REDUCE (FILE "Container1" 2000)
```

Anpassung von PREFETCHSIZE beim Hinzufügen oder Löschen von Containern

Die Standardgröße für alle Vorablesezugriffe von der Platte wird automatisch für alle Tabellenbereiche festgelegt, die mit DB2-Versionen ab Version 8.2 erstellt wurden. Dies bedeutet, dass der Datenbankmanager eine geeignete Vorablesezugriffsgröße berechnet und dabei eine Reihe von Faktoren als Basis verwendet, wie z. B. die Bereichsgröße, die Anzahl der Container im Tabellenbereich und die Eigenschaften Ihrer Speichereinheiten.

Der Grad, zu dem Vorablesezugriffe für Daten parallel stattfinden können, ist eine Funktion der Anzahl von Containern in einem Tabellenbereich sowie weiterer Faktoren. Wenn beispielsweise zwei oder mehr Container vorhanden sind, können die Vorablesezugriffe jedes Containers parallel stattfinden, wodurch die Gesamtleistung der Datenbank verbessert werden kann. Wenn die Anzahl von Containern in

einem Tabellenbereich durch Hinzufügen oder Löschen von Containern geändert wird, ändert sich möglicherweise auch die Datenmenge, für die ein effizienter Vorablesezugriff möglich ist. Beispiel: Wenn ein Container hinzugefügt wird, die Anzahl der vorab gelesenen Bereiche jedoch unverändert bleibt, nutzen Sie möglicherweise nicht die Gelegenheit, zusätzliche Daten aus dem neuen Container parallel zu den Daten aus den anderen Containern abzurufen. Wenn Container hinzugefügt oder gelöscht werden, kann durch das entsprechende Anpassen der Vorablesezugriffsgröße das Leistungsniveau beibehalten oder erhöht werden, indem Ein-/Ausgabeoperationen effizienter ausgeführt werden.

Sie können die Vorablesezugriffsgröße für Tabellenbereiche manuell festlegen, in diesem Fall müssen Sie jedoch sicherstellen, dass Sie den Wert entsprechend aktualisieren, wenn Sie an den Containern in Ihrem Tabellenbereich Änderungen vornehmen und Sie eine optimale Vorablesezugriffsleistung beibehalten möchten. Um die Vorablesezugriffsgröße nicht manuell aktualisieren zu müssen, können Sie als Wert für PREFETCHSIZE für den Tabellenbereich `AUTOMATIC` festlegen, wenn Sie die Anweisung CREATE TABLESPACE oder ALTER TABLESPACE verwenden. AUTOMATIC ist der Standardwert für PREFETCHSIZE, es sei denn, Sie haben den Standardwert für den Konfigurationsparameter **dft_prefetch_sz** modifiziert.

Wenn Sie die Vorablesezugriffsgröße manuell angeben möchten,habe Sie drei Möglichkeiten:

- Erstellen Sie den Tabellenbereich mit einer bestimmten Vorablesezugriffsgröße. Wenn Sie einen Wert für die Vorablesezugriffsgröße manuell auswählen, müssen Sie daran denken, die Vorablesezugriffsgröße anzupassen, sobald die Anzahl der Container, die dem Tabellenbereich zugeordnet sind, geändert wird.
- Wenn für den Datenbankkonfigurationsparameter **dft_prefetch_sz** ein vom Standardwert `AUTOMATIC` abweichender Wert festgelegt ist, lassen Sie die Vorablesezugriffsgröße beim Erstellen des Tabellenbereichs weg. Der Datenbankmanager überprüft diesen Parameter, wenn kein expliziter Wert für PREFETCHSIZE bei der Erstellung eines Tabellenbereichs angegeben wird. Wenn ein anderer Wert als `AUTOMATIC` festgestellt wird, wird dieser als Standardwert für die Vorablesezugriffsgröße verwendet. Sie müssen daran denken, bei Bedarf den Wert für die Vorablesezugriffsgröße anzupassen, wenn die Anzahl der dem Tabellenbereich zugeordneten Container geändert wird.
- Ändern Sie den Wert für PREFETCHSIZE manuell mithilfe einer Anweisung ALTER TABLESPACE.

Wenn Sie die Vorablesezugriffsgröße manuell anpassen, geben Sie eine Größe an, die einem Plattenstripe entspricht, um eine optimale E/A-Parallelität zu erreichen. Verwenden Sie für die manuelle Berechnung der Vorablesezugriffsgröße die folgende Formel:

Anzahl_der_Container × *Anzahl_der_Platten_pro_Container* × *Bereichsgröße*

Beispiel: Angenommen, die Bereichsgröße für eine Datenbank beträgt 8 Seiten und es sind 4 Container vorhanden, von denen sich jeder auf einer einzelnen physischen Platte befindet. Wenn für die Vorablesezugriffsgröße $4 \times 1 \times 8 = 32$ definiert wird, ergibt dies eine Vorablesezugriffsgröße von insgesamt 32 Seiten. Diese 32 Seiten werden parallel aus jedem der 4 Container gelesen.

Wenn Sie über mehr als eine physische Platte pro Container verfügen, zum Beispiel, wenn die einzelnen Container aus einem RAID-Array bestehen, müssen Sie zur Optimierung der E/A-Parallelität sicherstellen, dass die Registrierdatenbankvariable **DB2_PARALLEL_IO** korrekt definiert ist. (Siehe „Parallele E/A für Tabellenbe-

reichscontainer, die mehrere physische Platten verwenden".) Wenn Sie Container hinzufügen oder löschen, müssen Sie bei einer manuell definierten Vorablesezugriffsgröße daran denken, den Wert so zu aktualisieren, dass er einer geeigneten Vorablesezugriffsgröße entspricht. Beispiel: Angenommen, von insgesamt 4 Containern befindet sich jeder in einem RAID-4+1-Array und die Registrierdatenbankvariable **DB2_PARALLEL_IO** wurde für parallele Vorablesezugriffe von jeder physischen Platte definiert. Darüber hinaus wird von einer Bereichsgröße von 8 Seiten ausgegangen. Zum Einlesen eines Bereichs pro Container legen Sie eine Vorablesezugriffsgröße von $4 \times 4 \times 8 = 128$ Seiten fest.

Parallele E/A für Tabellenbereichscontainer, die mehrere physische Platten verwenden

Bevor die Vorablesezugriffsanforderungen an die Vorablesewarteschlangen übergeben werden, werden sie auf der Basis der Containeranzahl in einem Tabellenbereich in eine Reihe kleinerer, paralleler Vorablesezugriffsanforderungen unterteilt. Die Registrierdatenbankvariable **DB2_PARALLEL_IO** wird dazu verwendet, die Parallelität von Vorablesezugriffsanforderungen manuell zu überschreiben. (Dies wird auch als *Parallelität des Tabellenbereichs* bezeichnet.) Wenn für **DB2_PARALLEL_IO** der Standardwert NULL angegeben wird, entspricht die Parallelität eines Tabellenbereichs der Anzahl der Container im Tabellenbereich. Wird diese Registrierdatenbankvariable aktiviert, definiert sie die Anzahl physischer Platten pro Container; die Parallelität eines Tabellenbereichs entspricht der Anzahl der Container multipliziert mit dem für die Registrierdatenbankvariable **DB2_PARALLEL_IO** angegebenen Wert. Beispiel: Wenn Sie über einen Container in Ihrem Tabellenbereich verfügen, der aus einem RAID-5-Plattenarray besteht, definieren Sie diesen Parameter so, dass aus der einzelnen Vorablesezugriffsanforderung, die der Datenbankmanager andernfalls ausführen würde, 5 parallele Vorablesezugriffsanforderungen werden. Wenn Sie über zwei Container verfügen, von denen sich jeder in einem RAID-10-Array befindet, können Sie diesen Parameter so definieren, dass aus den beiden Vorablesezugriffsanforderungen für jeden Container 20 Vorablesezugriffsanforderungen werden, eine für jede der 10 Platten, die dem jeweiligen Container zugeordnet sind.

Im Folgenden sind einige weitere Beispiele für den Einfluss der Registrierdatenbankvariablen **DB2_PARALLEL_IO** auf die Parallelität von Vorablesezugriffen dargestellt. Es wird davon ausgegangen, dass die Tabellenbereiche mit der Einstellung AUTOMATIC für die Vorablesezugriffsgröße definiert wurden.

- **DB2_PARALLEL_IO**=NULL
  - Der Vorablesezugriff auf die Tabellenbereichscontainer erfolgt parallel auf der Basis einer Kombination aus den folgenden Elementen:
    - Anzahl der Container in jedem Tabellenbereich
    - Größe, die für Vorablesezugriffe in der Anweisung CREATE TABLESPACE oder ALTER TABLESPACE sowie im Konfigurationsparameter **dft_prefetch_sz** angegeben wurde
  - Die Vorablesezugriffe werden nicht in kleinere Anforderungen pro Platte unterteilt. Wenn einem Container mehrere physische Platten zugeordnet sind, finden Vorablesezugriffe von den Platten für einen einzelnen Container nicht parallel statt.
  -
- **DB2_PARALLEL_IO**=*
  - Alle Tabellenbereiche verwenden die Standardanzahl an Spindeln (6) für jeden Container. Die Vorablesezugriffsgröße beträgt das Sechsfache, wenn die parallele E/A aktiviert ist.

– Für alle Tabellenbereiche wird die parallele E/A aktiviert. Die Vorablesezugriffsanforderung wird in mehrere kleinere Anforderungen unterteilt, jeweils auf den Wert für PREFETCHSIZE dividiert durch den Wert für EXTENTSIZE (bzw. auf die Anzahl Container multipliziert mit der Anzahl Spindeln).

- **DB2_PARALLEL_IO**=*:3
  – Alle Tabellenbereiche verwenden 3 als Anzahl Spindeln pro Container.
  – Für alle Tabellenbereiche wird die parallele E/A aktiviert.
- **DB2_PARALLEL_IO**=*:3,1:1
  – Alle Tabellenbereiche verwenden 3 als Anzahl Spindeln pro Container, mit Ausnahme von Tabellenbereich 1, der den Wert 1 verwendet.
  – Für alle Tabellenbereiche wird die parallele E/A aktiviert.

Konvertieren von Tabellenbereichen zur Verwendung von dynamischem Speicher

Sie können einige oder alle DMS-Tabellenbereiche (DMS, Database Managed Space, vom Datenbankmanager verwalteter Tabellenbereich) in einer Datenbank konvertieren, sodass dynamischer Speicher verwendet wird. Die Verwendung von dynamischem Speicher vereinfacht Ihnen die Aufgaben des Speichermanagements.

Vorbereitende Schritte

Stellen Sie sicher, dass die Datenbank über mindestens eine Speichergruppe verfügt. Führen Sie dazu eine Abfrage für SYSCAT.STOGROUPS aus und geben Sie die Anweisung CREATE STOGROUP ein, wenn die Ergebnismenge leer ist.

Vorgehensweise

Zur Konvertierung eines DMS-Tabellenbereichs zur Verwendung von dynamischem Speicher wählen Sie eine der drei folgenden Methoden aus:

- **Ändern Sie einen einzelnen Tabellenbereich.** Bei dieser Methode bleibt der Tabellenbereich online. Allerdings ist eine Neuverteilungsoperation erforderlich, die einige Zeit benötigt, um Daten aus den Containern ohne dynamischen Speicher in die neuen Container mit dynamischem Speicher zu versetzen.
  1. Geben Sie den Tabellenbereich an, den Sie zur Verwendung von dynamischem Speicher konvertieren möchten. Geben Sie an, welche Speichergruppe der Tabellenbereich verwenden soll. Geben Sie die folgende Anweisung ein:

     ```
     ALTER TABLESPACE tabber1 MANAGED BY AUTOMATIC STORAGE USING STOGROUP sg-medium
     ```

     Dabei ist tabber1 der Tabellenbereich und sg-medium die Speichergruppe, für die der Tabellenbereich definiert ist.
  2. Versetzen Sie die benutzerdefinierten Daten aus den alten Containern in die Speicherpfade der Speichergruppe sg_medium. Geben Sie dazu die folgende Anweisung ein:

     ```
     ALTER TABLESPACE tabber1 REBALANCE
     ```

     Anmerkung: Wenn Sie die Option REBALANCE jetzt nicht angeben und die Anweisung ALTER TABLESPACE später mit der Option REDUCE eingeben, werden die Container mit dynamischem Speicher entfernt. Um dieses Problem zu vermeiden, geben Sie die Anweisung ALTER TABLESPACE unter Angabe der Option REBALANCE ein.
  3. Verwenden Sie die folgende Anweisung, um den Fortschritt der REBALANCE-Operation zu überwachen:

     ```
     SELECT * from tabelle (MON_GET_REBALANCE_STATUS( 'tabber1', -2))
     ```

- **Verwenden Sie eine umgeleitete Restoreoperation.** Wenn die umgeleitete Restoreoperation in Bearbeitung ist, können Sie auf die momentan konvertierten Tabellenbereiche nicht zugreifen. Bei einem vollständigen umgeleiteten Restore für eine Datenbank besteht auf keinen der Tabellenbereiche Zugriff, bis die Recovery abgeschlossen ist.
  1. Führen Sie den Befehl **RESTORE DATABASE** aus, indem Sie den Parameter **REDIRECT** angeben. Wenn Sie einen einzelnen Tabellenbereich konvertieren wollen, geben Sie außerdem den Parameter **TABLESPACE** an:

     ```
     RESTORE DATABASE datenbankname TABLESPACE (tabellenbereichsname) REDIRECT
     ```
  2. Führen Sie den Befehl **SET TABLESPACE CONTAINERS** unter Angabe des Parameters **USING AUTOMATIC STORAGE** für jeden Tabellenbereich aus, den Sie konvertieren wollen:

     ```
     SET TABLESPACE CONTAINERS FOR tabellenbereichs-id USING AUTOMATIC STORAGE
     ```
  3. Führen Sie den Befehl **RESTORE DATABASE** erneut aus, indem Sie diesmal den Parameter **CONTINUE** angeben:

     ```
     RESTORE DATABASE datenbankname CONTINUE
     ```
  4. Führen Sie den Befehl **ROLLFORWARD DATABASE** aus, indem Sie die Parameter **TO END OF LOGS** und **AND STOP** angeben:

     ```
     ROLLFORWARD DATABASE datenbankname TO END OF LOGS AND STOP
     ```

  Bei Verwendung einer umgeleiteten Restoreoperation muss eine zusätzliche Anweisung ALTER TABLESPACE eingegeben werden, um die Datenbankkataloge mit der korrekten Speichergruppenzuordnung für den Tabellenbereich zu aktualisieren. Die Zuordnung zwischen Tabellenbereichen und Speichergruppen wird in den Systemkatalogtabellen aufgezeichnet und während der umgeleiteten Restoreoperation nicht aktualisiert. Die Anweisung ALTER TABLESPACE aktualisiert nur die Katalogtabellen; die zusätzliche Verarbeitungszeit für eine REBALANCE-Operation entfällt. Wenn die Anweisung ALTER TABLESPACE nicht eingegeben wird, kann sich dies auf die Abfrageleistung auswirken. Wenn Sie die Standardspeichergruppe für den Tabellenbereich während der umgeleiteten Restoreoperation geändert haben, müssen Sie den Befehl **RESTORE DATABASE** mit dem Parameter **USING STOGROUP** eingeben, um die Konsistenz sämtlicher Datenbankpartitionen und Systemkataloge zu gewährleisten.

Beispiel

Gehen Sie wie folgt vor, um den vom Datenbankmanager verwalteten Tabellenbereich *SALES* während einer umgeleiteten Restoreoperation zur Verwendung von dynamischem Speicher zu konvertieren:

1. Zum Konfigurieren einer umgeleiteten Restoreoperation für `testdb` geben Sie den folgenden Befehl ein:

   ```
   RESTORE DATABASE testdb REDIRECT
   ```
2. Geben Sie an, dass der Tabellenbereich *SALES* mit dynamischem Speicher verwaltet werden soll. Der Tabellenbereich *SALES* verfügt über den ID-Wert 5.

   ```
   SET TABLESPACE CONTAINERS FOR 5 USING AUTOMATIC STORAGE
   ```

   Anmerkung: Zur Bestimmung des ID-Werts eines Tabellenbereichs während einer umgeleiteten Restoreoperation verwenden Sie die Option GENERATE SCRIPT des Befehls **RESTORE DATABASE**.
3. Setzen Sie den folgenden Befehl ab, um mit der Restoreoperation fortzufahren:

   ```
   RESTORE DATABASE testdb CONTINUE
   ```
4. Aktualisieren Sie die Speichergruppeninformationen in den Katalogtabellen.

   ```
   CONNECT TO testdb
   ALTER TABLESPACE SALES MANAGED BY AUTOMATIC STORAGE
   ```

5. Wenn Sie die Speichergruppe für den Tabellenbereich während der umgeleiteten Restoreoperation geändert haben, geben Sie den folgenden Befehl ein:
   ```
   RESTORE DATABASE testdb USING STOGROUP sg_standard
   ```

Ändern von Tabellenbereichen mit dynamischem Speicher

Die Wartung von Tabellenbereichen mit dynamischem Speicher erfolgt großenteils automatisch. Die Änderungen, die Sie an Tabellenbereichen mit dynamischem Speicher vornehmen können, beschränken sich auf die Neuverteilung von Daten und die Verringerung der Gesamttabellenbereichsgröße.

Tabellenbereiche mit dynamischem Speicher verwalten die Zuordnung von Speicher automatisch, indem sie Container nach Bedarf bis zu den Begrenzungen, die durch die Speicherpfade festgelegt sind, erstellen und erweitern. Sie selbst können an Tabellenbereichen mit dynamischem Speicher nur die folgenden Operationen ausführen:

- Neuverteilung
- Freigabe nicht belegten Speichers durch Senken der oberen Grenze
- Verringern der Größe des Gesamttabellenbereichs
- Speichergruppe für einen Tabellenbereich mit dynamischem Speicher ändern

Sie können die Daten eines Tabellenbereichs mit dynamischem Speicher neu verteilen, indem Sie einen Speicherpfad zu einer Speichergruppe hinzufügen. Dies bewirkt, dass der Tabellenbereich unverzüglich mit der Verwendung des neuen Speicherpfads beginnt. In ähnlicher Weite sorgt die Neuverteilung beim Löschen eines Speicherpfads aus einer Speichergruppe dafür, dass Daten aus den Containern in den zu löschenden Speicherpfaden herausgenommen und auf die verbleibenden Container verteilt werden.

Das Hinzufügen neuer Speicherpfade bzw. das Löschen von Pfaden erfolgt auf der Speichergruppenebene. Zum Hinzufügen von Speicherpfaden zu einer Datenbank mit dynamischem Speicher verwenden Sie die Klausel ADD der Anweisung ALTER STOGROUP. Sie können eine Neuverteilung je nach Wunsch ausführen oder auch nicht. Wenn Sie keine Neuverteilung ausführen, werden die neuen Speicherpfade erst genutzt, wenn die zuvor vorhandenen Container bis zur Kapazitätsgrenze gefüllt sind. Wenn Sie eine Neuverteilung ausführen, werden die neu hinzugefügten Speicherpfade unverzüglich nutzbar.

Zum Löschen von Speicherpfaden verwenden Sie die Klausel DROP der Anweisung ALTER STORGOUP. Diese Aktion versetzt die Speicherpfade in den Status „Löschen anstehend”. Das Anwachsen von Containern in dem angegebenen Speicherpfad wird beendet. Bevor jedoch der Pfad vollständig aus der Datenbank entfernt werden kann, müssen Sie eine Neuverteilung aller Tabellenbereiche auf die Speicherpfade mithilfe der Klausel REBALANCE des Befehls ALTER TABLESPACE ausführen. Wenn ein Tabellenbereich für temporäre Tabellen Container in einem Speicherpfad im Status "Löschen anstehend" enthält, können Sie den Tabellenbereich entweder löschen und erneut erstellen oder die Datenbank neu starten, um den Container aus dem Speicherpfad zu entfernen.

Einschränkung: Sie können Daten in Tabellenbereichen mit dynamischem Speicher für temporäre Tabellen nicht neu verteilen. Die Neuverteilung wird nur für reguläre und große Tabellenbereiche mit dynamischem Speicher unterstützt.

Sie können den Speicher unterhalb der oberen Grenze eines Tabellenbereichs verfügbar machen, indem Sie die Klausel LOWER HIGH WATER MARK der Anwei-

sung ALTER TABLESPACE verwenden. Dies bewirkt, dass so viele Speicherbereiche wie möglich in nicht belegte Speicherbereiche weiter unten im Tabellenbereich versetzt werden. Die obere Grenze für den Tabellenbereich wird während des Prozesses herabgesetzt, wobei Container jedoch die Größe behalten, die sie hatten, bevor die Operation ausgeführt wurde.

Tabellenbereiche mit dynamischem Speicher können mithilfe der Option REDUCE der Anweisung ALTER TABLESPACE verkleinert werden. Wenn Sie einen Tabellenbereich mit dynamischem Speicher verkleinern, versucht der Datenbankmanager die obere Grenze für den Tabellenbereich zu senken und die Größe der Tabellenbereichscontainer zu verringern. Bei dem Versuch, die obere Grenze zu senken, löscht der Datenbankmanager möglicherweise leere Container und versetzt belegte Speicherbereiche in freie Speicherbereiche weiter am Anfang des Tabellenbereichs. Anschließend wird die Größe von Containern angepasst, sodass die Gesamtgröße des Speichers im Tabellenbereich gleich oder unwesentlich größer als die obere Grenze ist.

Freigeben von nicht belegtem Speicher in Tabellenbereichen mit dynamischem Speicher

Wenn Sie einen Tabellenbereich mit dynamischem Speicher verkleinern, versucht der Datenbankmanager die *obere Grenze* für den Tabellenbereich zu senken und die Größe der Tabellenbereichscontainer zu verringern. Bei dem Versuch, die obere Grenze zu senken, löscht der Datenbankmanager möglicherweise leere Container und versetzt belegte Speicherbereiche in freien Speicher näher am Anfang des Tabellenbereichs. Anschließend wird die Größe von Container angepasst, sodass die Gesamtgröße des Speichers im Tabellenbereich gleich oder unwesentlich größer als die obere Grenze ist.

Vorbereitende Schritte

Der Tabellenbereich muss ein mit DB2 Version 9.7 oder einer späteren Version erstellter Tabellenbereich mit dynamischem Speicher sein. Die Funktion des konsolidierbaren Speichers ist nicht in Tabellenbereichen verfügbar, die mit früheren Versionen des DB2-Produkts erstellt wurden. Mithilfe der Tabellenfunktion MON_GET_TABLESPACE lässt sich prüfen, welche Tabellenbereiche in einer Datenbank die Funktion des konsolidierbaren Speichers unterstützen.

Informationen zu diesem Vorgang

Sie können die Größe eines Tabellenbereichs mit dynamischem Speicher verringern, für den die Funktion des konsolidierbaren Speichers auf eine von verschiedenen möglichen Arten aktiviert wurde. Sie können den Datenbankmanager anweisen, den Tabellenbereich um folgende Beträge zu verkleinern:

- Den maximal möglichen Betrag
- Einen Betrag, den Sie in KB, MB, GB oder in Seiten angeben
- Einen Prozentsatz der aktuellen Größe des Tabellenbereichs

In jedem Fall versucht der Datenbankmanager, die Größe dadurch zu verringern, dass er Speicherbereiche in Richtung zum Anfang des Tabellenbereichs versetzt. Dies senkt, sofern genügend freier Speicher verfügbar ist, die obere Grenze des Tabellenbereichs. Wenn das Versetzen von Speicherbereichen abgeschlossen ist, wird die Größe des Tabellenbereichs auf die neue obere Grenze verringert.

Sie verwenden die Klausel REDUCE der Anweisung ALTER TABLESPACE, um die Tabellenbereichsgröße für einen Tabellenbereich mit dynamischem Speicher zu verringern. Sie können, wie oben aufgeführt, einen Betrag angeben, um den der Tabellenbereich verkleinert werden soll.

Anmerkung:

- Wenn Sie keinen Betrag angeben, um den der Tabellenbereich verkleinert werden soll, wird die Tabellenbereichsgröße so weit wie möglich verkleinert, ohne Speicherbereiche zu versetzen. Der Datenbankmanager versucht, die Größe der Container zu verringern, indem er zunächst EXTENTSIZE große Speicherbereiche freigibt, für die Löschoperationen anstehen. (Es ist möglich, dass einige Speicherbereiche mit dem Status „Löschen anstehend" aus Wiederherstellbarkeitsgründen nicht freigegeben werden können, sodass sie verbleiben.) Wenn sich die obere Grenze unter den freigegebenen Speicherbereichen befand, wird sie gesenkt. Andernfalls erfolgt keine Änderung an der oberen Grenze. Als Nächstes wird die Größe der Container angepasst, sodass die Gesamtgröße des Speichers im Tabellenbereich gleich oder unwesentlich größer als die obere Grenze ist. Diese Operation wird durch die Anweisung ALTER TABLESPACE allein mit der Klausel REDUCE ausgeführt.
- Wenn Sie nur die obere Grenze senken wollen, indem Sie die belegten Speicherbereiche weiter unten im Tabellenbereich konsolidieren, ohne Containeroperationen auszuführen, können Sie die Anweisung ALTER TABLESPACE mit der Klausel LOWER HIGH WATER MARK verwenden.
- Wenn sich eine Operation REDUCE oder LOWER HIGH WATER MARK in Ausführung befindet, können Sie sie mithilfe der Klausel REDUCE STOP bzw. LOWER HIGH WATER MARK STOP der Anweisung ALTER TABLESPACE stoppen. Alle Speicherbereiche, die versetzt wurden, werden festgeschrieben, die obere Grenze wird auf den neuen Wert herabgesetzt und die Größe von Containern wird an die neue obere Grenze angepasst.

Einschränkungen

- Sie können Speicher nur in Tabellenbereichen freigeben, die mit DB2 Version 9.7 oder späteren Versionen erstellt wurden.
- Wenn Sie die Klausel REDUCE oder die Klausel LOWER HIGH WATER MARK in der Anweisung ALTER TABLESPACE angeben, können Sie keine anderen Parameter angeben.
- Wenn sich der Speicherbereich, der die Seite enthält, die zurzeit als obere Grenze angegeben ist, im Status „Löschen anstehend" befindet, kann der Versuch, die obere Grenze durch Versetzen von Speicherbereichen zu senken, fehlschlagen. In diesem Fall wird die Nachricht ADM6008I protokolliert. Speicherbereiche im Status „Löschen anstehend" können aus Wiederherstellbarkeitsgründen nicht immer versetzt werden. Solche Speicherbereiche werden schließlich durch normale Datenbankwartungsprozesse freigegeben, sodass sie dann versetzt werden können.

Vorgehensweise

Gehen Sie wie folgt vor, um die Größe eines Tabellenbereichs mit dynamischem Speicher zu verringern:

1. Formulieren Sie eine Anweisung ALTER TABLESPACE, die eine Klausel REDUCE enthält.

   ```
   ALTER TABLESPACE tabellenbereichsname REDUCE reduce-klausel
   ```

2. Führen Sie die Anweisung ALTER TABLESPACE aus.

Beispiel

Beispiel 1: Verkleinern eines Tabellenbereichs mit dynamischem Speicher um den maximal möglichen Betrag

```
ALTER TABLESPACE TS1 REDUCE MAX
```

In diesem Fall wird das Schlüsselwort MAX in der Klausel REDUCE angegeben, um den Datenbankmanager anzuweisen, die maximal mögliche Anzahl von Speicherbereichen in Richtung zum Anfang des Tabellenbereichs zu versetzen.

Beispiel 2: Verkleinern eines Tabellenbereichs mit dynamischem Speicher um einen Prozentsatz der aktuellen Tabellenbereichsgröße

```
ALTER TABLESPACE TS1 REDUCE 25 PERCENT
```

Durch diese Anweisung wird versucht, die Größe des Tabellenbereichs TS1 nach Möglichkeit auf 75 % seiner ursprünglichen Größe zu verringern.

Szenarios: Hinzufügen und Entfernen von Speicher bei Tabellenbereichen mit dynamischem Speicher

Die drei Szenarios in diesem Abschnitt veranschaulichen, welche Auswirkungen das Hinzufügen und Entfernen von Speicherpfaden auf Tabellenbereiche mit dynamischem Speicher hat.

Nachdem Speicherpfade für Speichergruppen hinzugefügt oder entfernt wurden, können Sie mithilfe einer Neuverteilungsoperation (REBALANCE) einen oder mehrere Container für die neuen Speicherpfade erstellen oder Container aus den gelöschten Pfaden entfernen. Bei der Neuverteilung der Daten von Tabellenbereichen ist Folgendes zu beachten:

- Wenn der Datenbankmanager aus irgendeinem Grund entscheidet, dass keine Container hinzugefügt oder gelöscht werden müssen oder wenn Container aufgrund von „Speichermangel" nicht hinzugefügt werden konnten, empfangen Sie eine Warnung.
- Die Klausel REBALANCE muss für sich allein angegeben werden.
- Sie können Daten in Tabellenbereichen mit dynamischem Speicher für temporäre Tabellen nicht neu verteilen. Es können nur Daten in regulären und großen Tabellenbereichen mit dynamischem Speicher neu verteilt werden.
- Der Aufruf einer Neuverteilung ist eine protokollierte Operation, die bei einem Rollforward wiederholt wird (auch wenn sich das Speicherlayout unterscheiden kann).
- In Umgebungen mit partitionierten Datenbanken wird eine Neuverteilung in jeder Datenbankpartition eingeleitet, in der sich der Tabellenbereich befindet.
- Wenn Speicherpfade hinzugefügt oder gelöscht werden, werden Sie nicht gezwungen, eine Neuverteilung auszuführen. Tatsächlich können nachfolgende Speicherpfadoperationen im Lauf der Zeit ausgeführt werden, ohne eine Neuverteilungsoperation auszuführen. Wenn ein Speicherpfad gelöscht wird und sich im Status „Nicht im Gebrauch" befindet, wird er im Rahmen der ALTER STOGROUP-Operation unverzüglich gelöscht. Falls sich der Speicherpfad im Status „Im Gebrauch" befindet und gelöscht wird, jedoch keine Neuverteilung der Tabellenbereiche ausgeführt wird, wird der Speicherpfad (jetzt im Status „Löschen anstehend") nicht mehr zum Speichern weiterer Container bzw. Daten verwendet.

Szenario: Hinzufügen eines Speicherpfads und Neuverteilen der Daten in Tabellenbereichen mit dynamischem Speicher:

Dieses Szenario zeigt, wie Speicherpfade einer Speichergruppe hinzugefügt werden und wie eine Neuverteilungsoperation (REBALANCE) einen oder mehrere Container in den neuen Speicherpfaden erstellt.

Dieses Szenario geht von der Annahme aus, dass der Speichergruppe ein neuer Speicherpfad hinzugefügt werden soll und dass ein vorhandener Tabellenbereich über diesen neuen Pfad übergreifend gespeichert werden soll. Die E/A-Parallelität wird durch Hinzufügen eines neuen Containers in jedem der Stripe-Sets des Tabellenbereichs verbessert.

Verwenden Sie die Anweisung ALTER STOGROUP, um einer Speichergruppe einen neuen Speicherpfad hinzuzufügen. Anschließend verwenden Sie die Klausel REBALANCE in der Anweisung ALTER TABLESPACE, um dem neuen Speicherpfad Container zuzuordnen und die Daten aus den vorhandenen Containern auf die neuen Container neu zu verteilen. Die Anzahl und die Größe der zu erstellenden Container hängt von der Definition der aktuellen Stripe-Sets für den Tabellenbereich und von der Größe des freien Speicherplatzes in den neuen Speicherpfaden ab.

Abb. 27 veranschaulicht durch das "Vorher"- und "Nachher"-Layout eines neu verteilten Tabellenbereichs, wie ein Speicherpfad hinzugefügt wird:

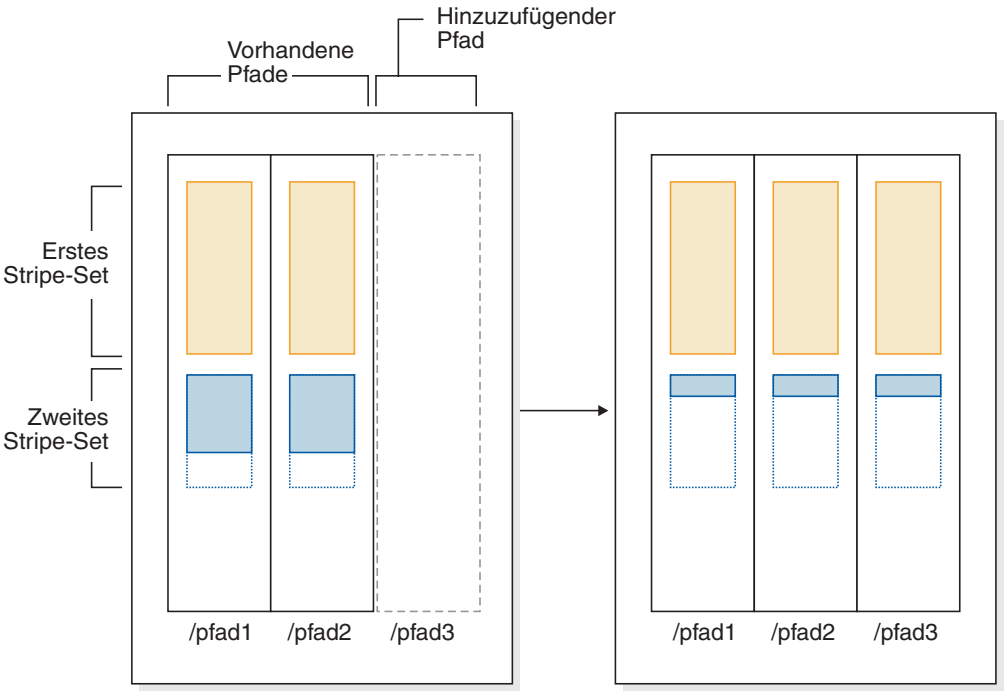


Abbildung 27. Hinzufügen eines Speicherpfads und Neuverteilen der Daten eines Tabellenbereichs mit dynamischem Speicher

Anmerkung: Die Diagramme in diesem Abschnitt dienen ausschließlich zu Veranschaulichungszwecken. Sie sind nicht dazu gedacht, eine bestimmte Strategie oder ein bewährtes Verfahren für das Speicherlayout nahezulegen. Darüber hinaus stellen die Diagramme nur einen Tabellenbereich dar. In der Praxis ist es jedoch eher wahrscheinlich, dass mehrere Tabellenbereiche mit dynamischem Speicher denselben Speicherpfad gemeinsam nutzen.

Zu einer ähnlichen Situation kann es kommen, wenn ein vorhandener Tabellenbereich mehrere Stripe-Sets mit unterschiedlichen Anzahlen von Containern hat, was

durch *vollen Plattenspeicher* in einem oder mehreren Speicherpfaden während der Lebensdauer des Tabellenbereichs verursacht werden kann. In diesem Fall ist es für den Datenbankmanager vorteilhaft, diesen vorhandenen Speicherpfaden Container hinzuzufügen, um die „Löcher" in den Stripe-Sets zu schließen (unter der Annahme natürlich, dass dazu jetzt freier Speicher verfügbar ist). Die REBALANCE-Operation kann auch dazu verwendet werden.

Abb. 28 zeigt ein Beispiel, bei dem ein „Loch" in den Stripe-Sets eines Tabellenbereichs, dessen Daten neu verteilt werden, (z. B. durch Löschen von Tabellenzeilen) entstanden ist. Dargestellt wird das „Vorher"- und „Nachher"-Layout der Speicherpfade.

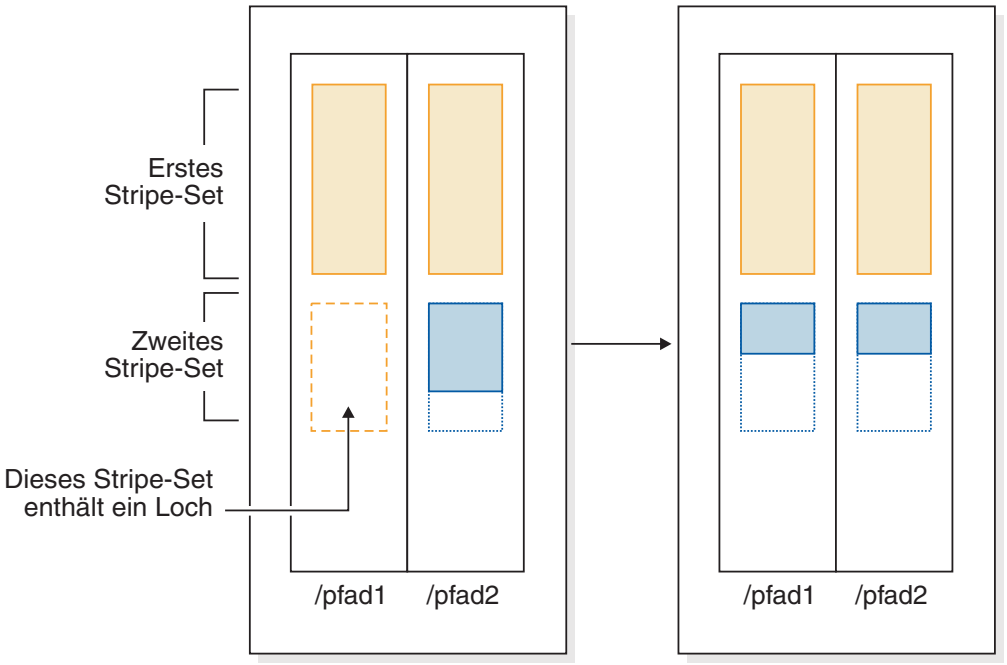


Abbildung 28. Neuverteilen der Daten eines Tabellenbereichs mit dynamischem Speicher zum Füllen von Löchern

Beispiel

Sie haben eine Speichergruppe mit zwei Speicherpfaden erstellt:

```
CREATE STOGROUP sg ON '/pfad1', '/pfad2'
```

Nach dem Erstellen der Datenbank wurden Tabellenbereiche mit dynamischem Speicher nachfolgend in dieser Speichergruppe erstellt.

Sie entscheiden sich, der Speichergruppe einen weiteren Speicherpfad (`/pfad3`) hinzuzufügen, und wollen, dass alle Tabellenbereiche mit dynamischem Speicher den neuen Speicherpfad verwenden.

1. Der erste Schritt besteht darin, den Speicherpfad der Speichergruppe hinzuzufügen:

   ```
   ALTER STOGROUP sg ADD '/pfad3'
   ```

2. Im nächsten Schritt ermitteln Sie alle betroffenen permanenten Tabellenbereiche. Dies geschieht durch ein manuelles Untersuchen der Ausgabe einer Tabellenbereichsmomentaufnahme oder durch SQL. Die folgende SQL-Anweisung generiert eine Liste aller regulären und großen Tabellenbereiche mit dynamischem Speicher in der Speichergruppe:

```
SELECT TBSP_NAME
   FROM table (MON_GET_TABLESPACE(' ', -2))
   WHERE TBSP_USING_AUTO_STORAGE = 1
      AND TBSP_CONTENT_TYPE IN ('ANY','LARGE')
      AND STORAGE_GROUP_NAME = 'sg'
   ORDER BY TBSP_ID
```

3. Wenn die Tabellenbereiche ermittelt wurden, besteht der nächste Schritt darin, die folgende Anweisung für jeden der aufgeführten Tabellenbereiche auszuführen. Vorausgesetzt, dass ausreichend Speicherplatz in den verbleibenden Speicherpfaden verfügbar ist, sollte es im Allgemeinen keine Rolle spielen, in welcher Reihenfolge die Neuverteilungen ausgeführt werden (und sie können parallel ausgeführt werden).

   ```
   ALTER TABLESPACE tabellenbereichsname REBALANCE
   ```

Anschließend müssen Sie festlegen, wie Tabellenbereiche für temporäre Tabellen gehandhabt werden sollen. Eine Option ist, die Datenbank zu stoppen (inaktivieren) und zu starten (aktivieren). Dies bewirkt, dass die Container erneut definiert werden. Alternativ können Sie die Tabellenbereiche für temporäre Tabellen löschen und erneut erstellen oder zuerst einen neuen Tabellenbereich für temporäre Tabellen erstellen und dann den alten löschen. Auf diese Weise vermeiden Sie den Versuch, den letzten Tabellenbereich für temporäre Tabellen in der Datenbank zu löschen, was nicht zulässig ist. Zur Ermittlung der Liste der betroffenen Tabellenbereiche können Sie die Ausgabe einer Tabellenbereichsmomentaufnahme manuell untersuchen oder Sie können eine SQL-Anweisung ausführen. Die folgende SQL-Anweisung generiert eine Liste aller Tabellenbereiche mit dynamischem Speicher für temporäre System- und Benutzertabellen in der Datenbank:

```
SELECT TBSP_NAME
   FROM table (MON_GET_TABLESPACE(' ', -2))
   WHERE TBSP_USING_AUTO_STORAGE = 1
      AND TBSP_CONTENT_TYPE IN ('USRTEMP','SYSTEMP')
      AND STORAGE_GROUP_NAME = 'sg'
   ORDER BY TBSP_ID
```

Szenario: Löschen eines Speicherpfads und Neuverteilen der Daten in Tabellenbereichen mit dynamischem Speicher:

Dieses Szenario zeigt, wie Speicherpfade gelöscht werden und wie die Operation REBALANCE Container aus den Tabellenbereichen löscht, die die Pfade verwenden.

Bevor die Operation zum Löschen eines Speicherpfads abgeschlossen werden kann, müssen alle Tabellenbereichscontainer in diesem Pfad entfernt werden. Wenn ein gesamter Tabellenbereich nicht mehr benötigt wird, können Sie ihn löschen, bevor Sie den Speicherpfad aus der Speichergruppe löschen. In diesem Fall ist keine Neuverteilung erforderlich. Wenn Sie den Tabellenbereich jedoch behalten möchten, ist eine Neuverteilungsoperation (REBALANCE) erforderlich. Wenn sich in diesem Fall Speicherpfade im Status „Löschen anstehend" befinden, führt der Datenbankmanager eine *regressive Neuverteilung* aus, bei der die Versetzung von Speicherbereichen mit dem Speicherbereich an der oberen Grenze beginnt (dem letztmöglichen Speicherbereich, der Daten im Tabellenbereich enthält) und mit dem Speicherbereich 0 endet.

Bei der Operation REBALANCE werden die folgenden Aktionen ausgeführt:

- Es wird eine regressive Neuverteilung ausgeführt. Daten in allen Containern im Status „Löschen anstehend" werden in die verbleibenden Container versetzt.
- Die Container im Status „Löschen anstehend" werden gelöscht.

- Wenn der aktuelle Tabellenbereich der letzte Tabellenbereich ist, der den Speicherpfad verwendet, wird der Speicherpfad ebenfalls gelöscht.

Wenn die Container in den verbleibenden Speicherpfaden nicht groß genug sind, um alle Daten, die versetzt werden, aufzunehmen, muss der Datenbankmanager möglicherweise zuerst Container in den verbleibenden Speicherpfaden erstellen oder erweitern, bevor die Neuverteilung ausgeführt werden kann.

Abb. 29 zeigt ein Beispiel für einen Speicherpfad, der gelöscht wird. Dargestellt wird das „Vorher"- und „Nachher"-Layout der Speicherpfade nach der Neuverteilung des Tabellenbereichs:

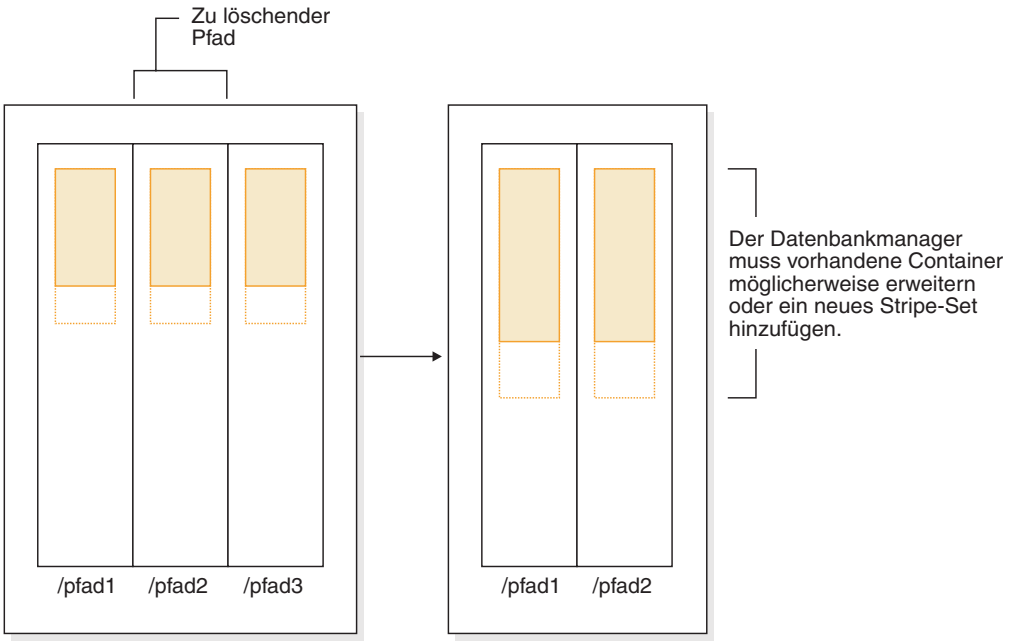


Abbildung 29. Löschen eines Speicherpfads und Neuverteilen eines Tabellenbereichs mit dynamischem Speicher

Beispiel

Erstellen Sie eine Speichergruppe mit drei Speicherpfaden:

```
CREATE STOGROUP sg ON '/pfad1', '/pfad2', '/pfad3'
```

Nach dem Erstellen der Speichergruppe wurden Tabellenbereiche mit dynamischem Speicher erstellt, die diese Speichergruppe verwenden.

Sie wollen den Speicherpfad /pfad3 in den Status "Löschen anstehend" versetzen, indem Sie ihn aus der Speichergruppe löschen und anschließend eine Neuverteilung aller Tabellenbereiche ausführen, die diesen Speicherpfad verwenden, sodass er gelöscht wird.

1. Der erste Schritt besteht darin, den Speicherpfad aus der Speichergruppe zu löschen:

   ```
   ALTER STOGROUP sg DROP '/pfad3'
   ```

2. Im nächsten Schritt ermitteln Sie alle betroffenen Tabellenbereiche für nicht temporäre Tabellen. Die folgende SQL-Anweisung generiert eine Liste aller regulären und großen Tabellenbereiche mit dynamischem Speicher in der Datenbank, die Container in einem Pfad mit dem Status „Löschen anstehend" haben:

```
SELECT TBSP_NAME
   FROM table (MON_GET_TABLESPACE(' ', -2))
   WHERE TBSP_USING_AUTO_STORAGE = 1
      AND TBSP_CONTENT_TYPE IN ('ANY','LARGE')
      AND STORAGE_GROUP_NAME = 'sg'
   ORDER BY TBSP_ID
```

3. Wenn die Tabellenbereiche ermittelt wurden, besteht der nächste Schritt darin, die folgende Anweisung für jeden der aufgeführten Tabellenbereiche auszuführen:

   ```
   ALTER TABLESPACE <tabellenbereichsname> REBALANCE
   ```

   a. Wenn Sie mehrere Speicherpfade aus der Speichergruppe gelöscht haben und den Speicher in einem bestimmten Pfad freigeben wollen, können Sie die Liste der Container in der Speichergruppe abfragen, um die Container zu ermitteln, die sich in dem Speicherpfad befinden. Betrachten Sie zum Beispiel einen Pfad mit dem Namen /pfad3. Die folgende Abfrage stellt eine Liste von Tabellenbereichen bereit, die Container im Pfad /pfad3 haben:

      ```
      SELECT TBSP_NAME FROM SYSIBMADM.SNAPCONTAINER
        WHERE CONTAINER_NAME LIKE '/pfad3'
        GROUP BY TBSP_NAME;
      ```

   b. Dann können Sie eine REBALANCE-Anweisung für jeden Tabellenbereich in der Ergebnismenge ausführen.

4. Generieren Sie eine Liste aller Tabellenbereiche mit dynamischem Speicher für temporäre System- und Benutzertabellen, die in den gelöschten Speicherpfaden definiert sind, um die Liste der betroffenen Tabellenbereiche zu ermitteln:

   ```
   SELECT TBSP_NAME
      FROM table (MON_GET_TABLESPACE(' ', -2))
      WHERE TBSP_USING_AUTO_STORAGE = 1
         AND TBSP_CONTENT_TYPE IN ('USRTEMP','SYSTEMP')
         AND STORAGE_GROUP_NAME = 'sg'
      ORDER BY TBSP_ID
   ```

Szenario: Hinzufügen und Entfernen von Speicherpfaden und Neuverteilen der Daten in Tabellenbereichen mit dynamischem Speicher:

Dieses Szenario zeigt, wie Speicherpfade hinzugefügt und entfernt werden können und wie die REBALANCE-Operation die Daten aller Tabellenbereiche mit dynamischem Speicher neu verteilt.

Es ist möglich, in einer Speichergruppe gleichzeitig Speicher hinzuzufügen und zu löschen. Diese Operation kann mithilfe nur einer Anweisung ALTER STOGROUP oder mithilfe mehrerer durch einen Zeitraum getrennter Anweisungen ALTER STOGROUP geschehen (wobei während dieser Zeit kein Neuausgleich für die Tabellenbereiche erfolgt).

Wie in „Szenario: Hinzufügen eines Speicherpfads und Neuverteilen der Daten in Tabellenbereichen mit dynamischem Speicher" auf Seite 255 beschrieben, kann es zu einer Situation kommen, in der der Datenbankmanager „Löcher" in Stripe-Sets ausfüllt, wenn Speicherpfade gelöscht werden. In diesem Fall erstellt der Datenbankmanager Container und löscht Container im Rahmen des Prozesses. In all diesen Szenarios erkennt der Datenbankmanager, dass einige Container hinzugefügt (wo es freier Speicherplatz zulässt) und einige entfernt werden müssen. In diesen Szenarios muss der Datenbankmanager möglicherweise eine Neuverteilungsoperation in zwei Durchgängen ausführen (Phase und Status dieser Operation werden in der Ausgabe des Momentaufnahmemonitors beschrieben):

1. Zuerst werden neue Container in den neuen Pfaden zugeordnet (bzw. in vorhandenen Pfaden, wenn „Löcher" aufgefüllt werden).

2. Eine progressive Neuverteilung wird ausgeführt.
3. Eine regressive Neuverteilung wird ausgeführt, wobei Daten aus den Containern in den Pfaden, die gelöscht werden, entfernt und in andere Container versetzt werden.
4. Die Container werden physisch gelöscht.

Abb. 30 zeigt ein Beispiel für Speicherpfade, die hinzugefügt und gelöscht werden. Dargestellt wird das "Vorher"- und "Nachher"-Layout eines neu verteilten Tabellenbereichs:

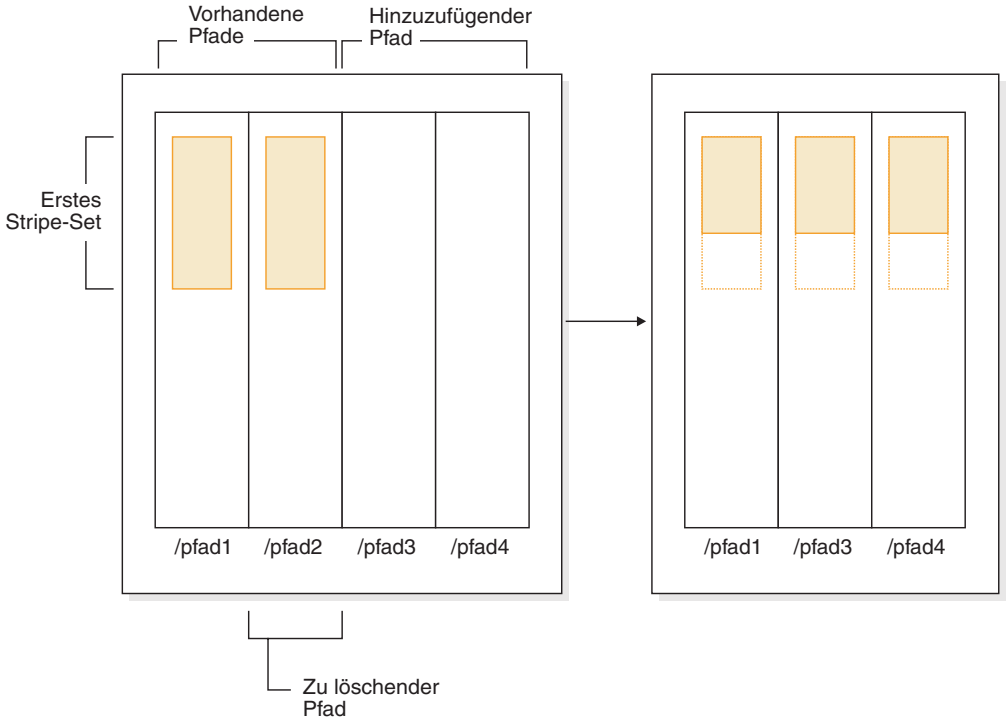


Abbildung 30. Hinzufügen und Löschen von Speicherpfaden mit anschließendem Neuverteilen der Daten in einem Tabellenbereich mit dynamischem Speicher

Beispiel

Eine Datenbank wird mit zwei Speicherpfaden erstellt:

```
CREATE STOGROUP sg ON '/pfad1', '/pfad2', '/pfad4'
```

Nehmen Sie an, Sie wollen der Speichergruppe einen weiteren Speicherpfad (`/pfad3`) hinzufügen und einen der vorhandenen Pfade (`/pfad2`) entfernen. Weiterhin sollen die Daten aller Tabellenbereiche mit dynamischem Speicher neu verteilt werden. Der erste Schritt besteht darin, den neuen Speicherpfad `/pfad3` der Speichergruppe hinzuzufügen und das Entfernen des Speicherpfads `/pfad2` einzuleiten:

```
ALTER STOGROUP sg ADD '/pfad3'
ALTER STOGROUP sg DROP '/pfad2'
```

Im nächsten Schritt ermitteln Sie alle betroffenen Tabellenbereiche. Diese Analyse geschieht durch ein manuelles Untersuchen der Ausgabe einer Tabellenbereichsmomentaufnahme oder durch SQL-Anweisungen. Die folgende SQL-Anweisung generiert eine Liste aller regulären und großen Tabellenbereiche mit dynamischem Speicher in der Datenbank:

```
SELECT TBSP_NAME
   FROM table (MON_GET_TABLESPACE(' ', -2))
   WHERE TBSP_USING_AUTO_STORAGE = 1
      AND TBSP_CONTENT_TYPE IN ('ANY','LARGE')
      AND STORAGE_GROUP_NAME = 'sg'
   ORDER BY TBSP_ID
```

Wenn die Tabellenbereiche ermittelt wurden, besteht der nächste Schritt darin, die folgende Anweisung für jeden der aufgeführten Tabellenbereiche auszuführen:

```
ALTER TABLESPACE tabellenbereichsname REBALANCE
```

Dabei ist *tabellenbereichsname* der Name der Tabellenbereiche, die im vorherigen Schritt ermittelt wurden.

Anmerkung: Sie können Daten in Tabellenbereichen für temporäre Tabellen, die durch die Funktion für dynamischen Speicher verwaltet werden, nicht neu verteilen. Wenn Sie die Verwendung des Speichers, der Tabellenbereichen für temporäre Tabellen zugeordnet wurde, beenden wollen, besteht eine Option darin, die Tabellenbereiche für temporäre Tabellen zu löschen und anschließend erneut zu erstellen.

Zuordnen eines Tabellenbereichs zu einer Speichergruppe

Mit der Anweisung CREATE TABLESPACE oder der Anweisung ALTER TABLESPACE können Sie die Speichergruppe angeben oder ändern, die von einem Tabellenbereich verwendet wird. Wenn bei der Erstellung eines Tabellenbereichs keine Speichergruppe angegeben wird, wird die Standardspeichergruppe verwendet.

Informationen zu diesem Vorgang

Wenn Sie die von einem Tabellenbereich verwendete Speichergruppe ändern, wird beim Festschreiben der Anweisung ALTER TABLESPACE eine implizite REBALANCE-Operation ausgeführt. Diese versetzt die Daten aus der Quellenspeichergruppe in die Zielspeichergruppe.

Bei Verwendung von IBM DB2 pureScale Feature wird REBALANCE nicht unterstützt und die zugeordnete Speichergruppe kann nicht geändert werden. Die REBALANCE-Operation ist asynchron und hat keine Auswirkungen auf die Datenverfügbarkeit. Mit der Monitortabellenfunktionen MON_GET_REBALANCE_STATUS kann der Fortschritt der REBALANCE-Operation überwacht werden.

Während der Operation ALTER TABLESPACE werden kompilierte Objekte, die auf alten Tabellenbereichsattributen basieren, *vorläufig inaktiviert*. Neue Kompilierungen nach der Festschreibung von ALTER TABLESPACE verwenden die in der Anweisung ALTER TABLESPACE angegebenen neuen Tabellenbereichsattribute. Die Unterstützung für *vorläufige Inaktivierung* ist auf dynamisches SQL beschränkt. Statische SQL-Abhängigkeiten müssen für die neuen Werte manuell erkannt und erneut kompiliert werden.

Tabellenbereiche, die dieselbe Speichergruppe verwenden, können unterschiedliche Werte für PAGESIZE und EXTENTSIZE aufweisen. Diese Attribute beziehen sich auf die Tabellenbereichsdefinition und nicht auf die Speichergruppe.

Vorgehensweise

Geben Sie die folgende Anweisung ein, um einen Tabellenbereich einer Speichergruppe zuzuordnen:

```
CREATE TABLESPACE tbber USING STOGROUP
speichergruppe
```

Dabei ist *tbber* der neue Tabellenbereich und *speichergruppe* die zugeordnete Speichergruppe.

Szenario: Versetzen eines Tabellenbereichs in eine neue Speichergruppe

Das folgende Szenario zeigt, wie ein Tabellenbereich aus einer Speichergruppe in eine andere Speichergruppe versetzt werden kann.

In diesem Szenario wird davon ausgegangen, dass sich die Tabellenbereichsdaten in Containern befinden, die in den Speicherpfaden einer Speichergruppe enthalten sind. Die Tabellenbereichsdaten werden mit der Anweisung ALTER TABLESPACE in die neue Speichergruppe versetzt.

Beim Versetzen des Tabellenbereichs in die neue Speichergruppe werden die Container in der alten Speichergruppe in den Status 'Löschen anstehend' versetzt. Nach dem Festschreiben der Anweisung ALTER TABLESPACE werden die Container den Speicherpfaden der neuen Speichergruppe zugeordnet, die in den alten Speichergruppen enthaltenen Container werden in den Status 'Löschen anstehend' versetzt, und eine implizite REBALANCE-Operation wird eingeleitet. Mit dieser Operation werden dem neuen Speicherpfad Container zugeordnet und die Daten aus den vorhandenen Containern auf die neuen Container neu verteilt. Die Anzahl und die Größe der zu erstellenden Container hängt von der Anzahl der Speicherpfade in der Zielspeichergruppe und der Menge des freien Speicherplatzes in den neuen Speicherpfaden ab. Nach dem Versetzen sämtlicher Daten werden die alten Container gelöscht.

Das folgende Diagramm stellt beispielhaft dar, wie ein Tabellenbereich aus einer Speichergruppe in eine andere Speichergruppe versetzt wird. Dabei werden die folgenden Aktionen ausgeführt:

1. Den Speicherpfaden der Zielspeichergruppe werden neue Container zugeordnet.
2. Alle ursprünglichen Container werden in den Status 'Löschen anstehend' versetzt; neue Zuordnungsanforderungen werden von den neuen Containern verarbeitet.
3. Eine regressive Neuverteilung wird ausgeführt, wobei Daten aus den Containern in den Pfaden, die gelöscht werden, entfernt und in andere Container versetzt werden.
4. Die Container werden physisch gelöscht.

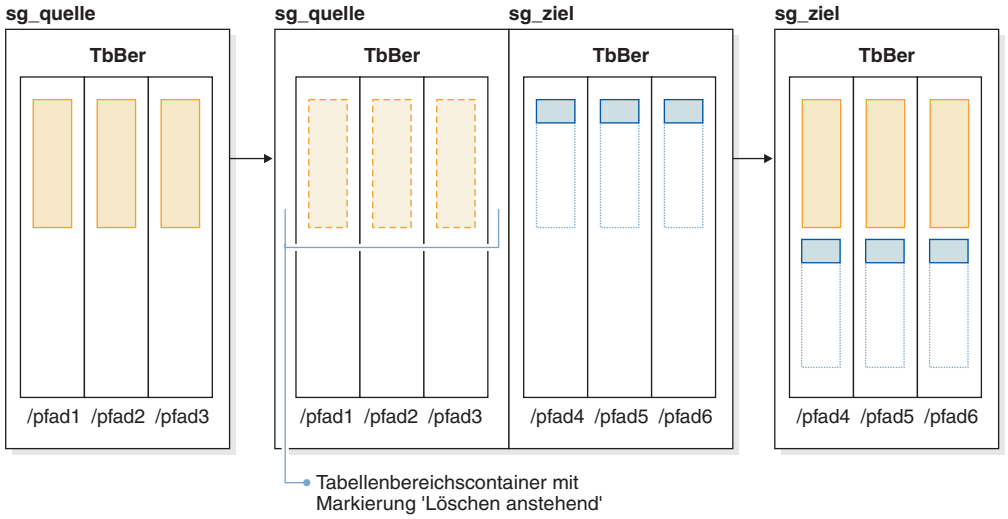


Abbildung 31. Versetzen eines Tabellenbereichs in eine neue Speichergruppe

Gehen Sie wie folgt vor, um einen Tabellenbereich in eine andere Speichergruppe zu versetzen:

1. Erstellen Sie zwei Speichergruppen, 'sg_quelle' und 'sg_ziel':
   ```
   CREATE STOGROUP sg_quelle ON '/pfad1', '/pfad2', '/pfad3'
   CREATE STOGROUP sg_ziel ON '/pfad4', '/pfad5', '/pfad6'
   ```
2. Erstellen Sie nach der Erstellung der Datenbank einen Tabellenbereich mit dynamischem Speicher, der anfänglich die Speichergruppe 'sg_quelle' verwendet:
   ```
   CREATE TABLESPACE tbber USING STOGROUP sg_quelle
   ```
3. Versetzen Sie den Tabellenbereich mit dynamischem Speicher in die Speichergruppe 'sg_ziel':
   ```
   ALTER TABLESPACE tbber USING sg_ziel
   ```

Umbenennen eines Tabellenbereichs

Zum Umbenennen eines Tabellenbereichs verwenden Sie die Anweisung RENAME TABLESPACE.

Informationen zu diesem Vorgang

Der Tabellenbereich SYSCATSPACE kann nicht umbenannt werden. Tabellenbereiche, die sich im Status 'Aktualisierende Recovery anstehend' oder 'Aktualisierende Recovery wird ausgeführt' befinden, können nicht umbenannt werden.

Beim Restore eines Tabellenbereichs, der seit seinem Backup umbenannt wurde, müssen Sie im Befehl **RESTORE DATABASE** den neuen Tabellenbereichsnamen verwenden. Wenn Sie den vorherigen Tabellenbereichsnamen benutzen, kann der Tabellenbereich nicht lokalisiert werden. Wenn Sie für den Tabellenbereich mit dem Befehl **ROLLFORWARD DATABASE** eine aktualisierende Recovery durchführen, müssen Sie ebenfalls den neuen Namen verwenden. Wenn Sie den vorherigen Tabellenbereichsnamen benutzen, kann der Tabellenbereich nicht lokalisiert werden.

Sie können einem vorhandenen Tabellenbereich einen neuen Namen zuordnen, ohne dass sich dies auf die einzelnen Objekte innerhalb des Tabellenbereichs auswirkt. Beim Umbenennen eines Tabellenbereichs werden alle Katalogsätze geändert, die auf diesen Tabellenbereich verweisen.

Statuswerte für Tabellenbereiche

In diesem Thema finden Sie Informationen zu den unterstützten Statuswerten für Tabellenbereiche.

Es gibt aktuell mindestens 25 Statuswerte für Tabellen oder Tabellenbereiche, die von dem IBM DB2-Datenbankprodukt unterstützt werden. Diese Statuswerte werden verwendet, um den Zugriff auf Daten unter bestimmten Umständen zu steuern, oder, falls erforderlich, um bestimmte Benutzeraktionen zu sondieren, damit die Integrität der Datenbank geschützt bleibt. Die meisten Statuswerte sind das Ergebnis von Ereignissen, die in Bezug zur Operation eines der DB2-Datenbankdienstprogramme stehen, wie z. B. das Dienstprogramm LOAD oder die Dienstprogramme zum Sichern und Wiederherstellen. In der folgenden Tabelle werden die einzelnen unterstützten Statuswerte für Tabellenbereiche beschrieben. Die Tabelle stellt Ihnen auch Arbeitsbeispiele zur Verfügung, an denen Sie genau sehen, wie Statuswerte, die beim Verwalten Ihrer Datenbank auftreten, interpretiert werden und wie Sie auf sie antworten. Die Beispiele stammen aus Befehlsscripts, die unter AIX ausgeführt wurden. Sie können Sie kopieren, einfügen und selbst ausführen. Wenn Sie das DB2-Datenbankprodukt unter einem anderen System als UNIX ausführen, müssen Sie sicherstellen, dass die Pfadnamen in dem für Ihr System korrekten Format sind. Die meisten Beispiele basieren auf Tabellen aus der Datenbank SAMPLE, die im Lieferumfang des DB2-Datenbankprodukts enthalten ist. Einige Beispiele erfordern Szenarios, die nicht Bestandteil der Datenbank SAMPLE sind, aber Sie können eine Verbindung zur Datenbank SAMPLE als Startpunkt verwenden.

Tabelle 17. Unterstützte Statuswerte für Tabellenbereiche

Status	Hexadezimaler Statuswert	Beschreibung	Beispiele

Backup anstehend	0x20	Diesen Status weist ein Tabellenbereich nach einer aktualisierenden Recovery für einen Tabellenbereich mit Zeitangabe oder nach einer Ladeoperation (für eine wiederherstellbare Datenbank) auf, für die die Option **COPY NO** angegeben ist. Bevor der Tabellenbereich verwendet werden kann, muss für den Tabellenbereich (oder alternativ für die gesamte Datenbank) ein Backup durchgeführt werden. Wenn für die Tabelle kein Backup durchgeführt wird, können Tabellen in diesem Tabellenbereich abgefragt, aber nicht aktualisiert werden. **Anmerkung:** Darüber hinaus muss sofort nach der Aktivierung für die aktualisierende Recovery ein Backup für die Datenbank durchgeführt werden. Eine Datenbank ist wiederherstellbar, wenn für den Datenbankkonfigurationsparameter **logretain** der Wert RECOVERY eingestellt ist oder wenn der Datenbankkonfigurationsparameter **userexit** auf den Wert YES eingestellt ist. Sie können eine solche Datenbank erst aktivieren oder eine Verbindung zu ihr herstellen, wenn für sie ein Backup durchgeführt wurde, bei dem der Wert des Datenbankkonfigurationsparameters **backup_pending** (der der Information dient) auf den Wert NO gesetzt wird.	1. Eingabedatei zum Laden 'staff_data.del' mit dem Inhalt: 11,"Melnyk",20,"Sales",10,70000,15000:
			`update db cfg for sample using logretain recovery;`
			`backup db sample;`
			`connect to sample;`
			`load from staff_data.del of del messages load.msg insert into staff copy no;`
			`update staff set salary = 69000 where id = 11;`
			2. `update db cfg for sample using logretain recovery;`
			`connect to sample;`

Tabelle 17. Unterstützte Statuswerte für Tabellenbereiche (Forts.)

Status	Hexadezimaler Statuswert	Beschreibung	Beispiele

Backup wird ausgeführt	0x800	Dies ist ein transienter Status, der nur während einer Backup-Operation wirksam ist.	Setzen Sie online den Befehl **BACKUP DATABASE** ab:
			`backup db sample online;`
			
			Führen Sie während der Ausführung der Backup-Operation eines der folgenden Scripts in einer anderen Sitzung aus:
			`connect to sample;`
			• `list tablespaces show detail;`
			• `get snapshot for tablespaces on sample;`
			`connect reset;`
			
			Die für USERSPACE1 zurückgegebenen Informationen zeigen an, dass dieser Tabellenbereich sich im Status 'Backup wird ausgeführt' befindet.

DMS-Ausgleichsfunktion ist aktiv	0x10000000	Dies ist ein transienter Status, der nur während einer Operation zum Ausgleich von Daten wirksam ist. Wenn einem Tabellenbereich, der als vom Datenbankmanager verwalteter Tabellenbereich (DMS) definiert ist, neue Container hinzugefügt oder vorhandene Container erweitert werden, erfolgt möglicherweise ein Ausgleich der Tabellenbereichsdaten. Der *Ausgleich* ist der Prozess zum Versetzen von Speicherbereichen von einer Position an eine andere, um die verteilte Speicherung (Striping) der Daten beizubehalten. Ein *Speicherbereich* ist eine Containerbereichseinheit (gemessen in Seiten) und ein 'Stripe' ist eine Speicherbereichsschicht für einen Tabellenbereich, die *einheitenübergreifend über die Gruppe von Containern* verteilt ist.	Eingabedatei `staffdata.del` zum Laden mit einem erheblichen Datenvolumen (z. B. 20000 oder mehr Datensätze):
			`connect to sample;`
			`create tablespace ts1 managed by database using (file '/home/melnyk/melnyk/NODE0000/SQL00001 /ts1c1' 1024);`
			`create table newstaff like staff in ts1;`
			`load from staffdata.del of del insert into newstaff nonrecoverable;`
			`alter tablespace ts1 add (file '/home/melnyk/melnyk /NODE0000/SQL00001/ts1c2' 1024);`
			`list tablespaces;`
			`connect reset;`
			
			Die für TS1 zurückgegebenen Informationen zeigen an, dass dieser Tabellenbereich sich im Status 'DMS-Ausgleichsfunktion ist aktiv' befindet.

Tabelle 17. Unterstützte Statuswerte für Tabellenbereiche (Forts.)

Status	Hexadezimaler Statuswert	Beschreibung	Beispiele
Inaktivierung anstehend	0x200	In diesem Status kann sich ein Tabellenbereich während der aktualisierenden Recovery für eine Datenbank befinden. Bei Beendigung der aktualisierenden Recovery sollte sich der Tabellenbereich nicht mehr in diesem Status befinden. Der Status wird durch Bedingungen ausgelöst, bei denen ein Tabellenbereich in den Offline-Status versetzt wurde und keine ausgleichenden Protokollsätze für eine Transaktion geschrieben wurden. Das Auftreten und nachfolgende Verschwinden dieses Tabellenbereichsstatus ist für Benutzer transparent.	Ein Beispiel für diesen Tabellenbereichsstatus würde den Rahmen dieses Dokuments sprengen.
Löschen anstehend	0x8000	In diesem Status befindet sich ein Tabellenbereich, wenn während einer Operation zum Neustart der Datenbank mindestens bei einem der zugehörigen Container ein Problem aufgetreten ist. (Eine Datenbank muss erneut gestartet werden, wenn die vorherige Sitzung mit dieser Datenbank abnormal beendet wurde, wie z. B. bei einem Stromausfall). Wenn sich ein Tabellenbereich im Status 'Löschen anstehend' befindet, ist er nicht verfügbar und kann nur gelöscht werden.	Ein Beispiel für diesen Tabellenbereichsstatus würde den Rahmen dieses Dokuments sprengen.

Tabelle 17. Unterstützte Statuswerte für Tabellenbereiche (Forts.)

Status	Hexadezimaler Statuswert	Beschreibung	Beispiele

Laden wird ausgeführt	0x20000	Dies ist ein transienter Status, der nur während einer Ladeoperation (für eine wiederherstellbare Datenbank) wirksam ist, bei der die Option **COPY NO** angegeben ist. Siehe auch den Tabellenstatus 'Ladevorgang wird ausgeführt'.	Eingabedatei `staffdata.del` zum Laden mit einem erheblichen Datenvolumen (z. B. 20000 oder mehr Datensätze):
			`update db cfg for sample using logretain recovery;`
			`backup db sample;`
			`connect to sample;`
			`create table newstaff like staff;`
			`load from staffdata.del of del insert into newstaff copy no;`
			`connect reset;`
			
			Führen Sie während der Ausführung der Ladeoperation das folgende Script in einer anderen Sitzung aus:
			`connect to sample;`
			`list tablespaces;`
			`connect reset;`
			
			Die für USERSPACE1 zurückgegebenen Informationen zeigen an, dass dieser Tabellenbereich sich im Status 'Ladevorgang wird ausgeführt' (und 'Backup anstehend') befindet.

Normal	0x0	Ein Tabellenbereich weist den Status 'Normal' auf, wenn er nicht einen der übrigen (abnormalen) Tabellenbereichsstatus aufweist. Der Status 'Normal' ist der Anfangsstatus eines Tabellenbereichs nach Erstellung.	`connect to sample;`
			`create tablespace ts1 managed by database using (file '/home/melnyk/melnyk/NODE0000/SQL00001 /tsc1' 1024);`
			`list tablespaces show detail;`

Tabelle 17. Unterstützte Statuswerte für Tabellenbereiche (Forts.)

Status	Hexadezimaler Statuswert	Beschreibung	Beispiele

Offline und kein Zugriff	0x4000	In diesem Status befindet sich ein Tabellenbereich, wenn bei mindestens einem der zugehörigen Container ein Program aufgetreten ist. Ein Container kann versehentlich umbenannt, versetzt oder beschädigt werden. Wenn das Problem behoben ist und auf die dem Tabellenbereich zugeordneten Container wieder zugegriffen werden kann, kann dieser abnormale Status gelöscht werden, indem alle Anwendungen von der Datenbank getrennt und anschließend erneut mit der Datenbank verbunden werden. Alternativ dazu können Sie die Anweisung ALTER TABLESPACE mit der Klausel SWITCH ONLINE ausgeben, um den Status 'Offline und kein Zugriff' aus dem Tabellenbereich zu löschen, ohne andere Anwendungen von der Datenbank zu trennen.	`connect to sample;`
			`create tablespace ts1 managed by database using (file '/home/melnyk/melnyk/NODE0000/SQL00001 /tsc1' 1024);`
			`alter tablespace ts1 add (file '/home/melnyk/melnyk /NODE0000/SQL00001/tsc2' 1024);`
			`export to st_data.del of del select * from staff;`
			`create table stafftemp like staff in ts1;`
			`import from st_data.del of del insert into stafftemp;`
			`connect reset;`
			
			Benennen Sie den Tabellenbereichscontainer 'tsc1' in 'tsc3' um und versuchen Sie anschließend, eine Abfrage für die Tabelle STAFFTEMP auszuführen:
			`connect to sample;`
			`select * from stafftemp;`
			
			Die Abfrage gibt SQL0290N (Tabellenbereichszugriff nicht zulässig) zurück und der Befehl **LIST TABLESPACES** gibt für TS1 den Statuswert 0x4000 (Offline und kein Zugriff) zurück. Benennen Sie den Tabellenbereichscontainer 'tsc3' wieder in 'tsc1' um. Dieses Mal wird die Abfrage erfolgreich ausgeführt.

Quiesce: EXCLUSIVE	0x4	In diesem Status befindet sich ein Tabellenbereich, wenn die Anwendung, die die Quiescefunktion für den Tabellenbereich aufruft, über ausschließlichen Zugriff (Lese- oder Schreibzugriff) auf den Tabellenbereich verfügt. Sie können einen Tabellenbereich explizit in den Status 'Quiesce: EXCLUSIVE' versetzen, indem Sie den Befehl **QUIESCE TABLESPACES FOR TABLE** absetzen.	Stellen Sie sicher, dass der Tabellenbereich den Status 'Normal' aufweist, bevor er in den Status 'Quiesce: EXCLUSIVE' versetzt wird.
			`connect to sample;`
			`quiesce tablespaces for table staff reset;`
			`quiesce tablespaces for table staff exclusive;`
			`connect reset;`
			
			Führen Sie das folgende Script in einer anderen Sitzung aus:
			`connect to sample;`
			`select * from staff where id=60;`
			`update staff set salary=50000 where id=60;`
			`list tablespaces;`
			`connect reset;`
			
			Die für USERSPACE1 zurückgegebenen Informationen zeigen an, dass sich dieser Tabellenbereich im Status 'Quiesce: EXCLUSIVE' befindet.

Tabelle 17. Unterstützte Statuswerte für Tabellenbereiche (Forts.)

Status	Hexadezimaler Statuswert	Beschreibung	Beispiele

Quiesce: SHARE	0x1	In diesem Status befindet sich ein Tabellenbereich, wenn sowohl die Anwendung, die die Quiescefunktion für den Tabellenbereich aufruft, als auch gleichzeitig ablaufende Anwendungen über Lesezugriff (aber nicht Schreibzugriff) auf den Tabellenbereich verfügen. Sie können einen Tabellenbereich explizit in den Status 'Quiesce: SHARE' versetzen, indem Sie den Befehl **QUIESCE TABLESPACES FOR TABLE** absetzen.	Stellen Sie sicher, dass der Tabellenbereich den Status 'Normal' aufweist, bevor er in den Status 'Quiesce: Share' versetzt wird.
			`connect to sample;`
			`quiesce tablespaces for table staff reset;`
			`quiesce tablespaces for table staff share;`
			`connect reset;`
			
			Führen Sie das folgende Script in einer anderen Sitzung aus:
			`connect to sample;`
			`select * from staff where id=40;`
			`update staff set salary=50000 where id=40;`
			`list tablespaces;`
			`connect reset;`
			
			Die für USERSPACE1 zurückgegebenen Informationen zeigen an, dass sich dieser Tabellenbereich im Status 'Quiesce: SHARE' befindet.

Quiesce: UPDATE	0x2	In diesem Status befindet sich ein Tabellenbereich, wenn die Anwendung, die die Quiescefunktion für den Tabellenbereich aufruft, über ausschließlichen Schreibzugriff auf den Tabellenbereich verfügt. Sie können einen Tabellenbereich explizit in den Status 'Quiesce: UPDATE' versetzen, indem Sie den Befehl **QUIESCE TABLESPACES FOR TABLE** absetzen.	Stellen Sie sicher, dass der Tabellenbereich den Status 'Normal' aufweist, bevor er in den Status 'Quiesce: UPDATE' versetzt wird.
			`connect to sample;`
			`quiesce tablespaces for table staff reset;`
			`quiesce tablespaces for table staff intent to update;`
			`connect reset;`
			
			Führen Sie das folgende Script in einer anderen Sitzung aus:
			`connect to sample;`
			`select * from staff where id=50;`
			`update staff set salary=50000 where id=50;`
			`list tablespaces;`
			`connect reset;`
			
			Die für USERSPACE1 zurückgegebenen Informationen zeigen an, dass sich dieser Tabellenbereich im Status 'Quiesce: UPDATE' befindet.

Tabelle 17. Unterstützte Statuswerte für Tabellenbereiche (Forts.)

Status	Hexadezimaler Statuswert	Beschreibung	Beispiele

Reorganisation wird ausgeführt	0x400	Dies ist ein transienter Status, der nur während einer Reorganisationsoperation wirksam ist.	Setzen Sie den Befehl **REORG TABLE** ab:
			`connect to sample;`
			`reorg table staff;`
			`connect reset;`
			
			Führen Sie während der Ausführung der Reorganisationsoperation eines der folgenden Scripts in einer anderen Sitzung aus:
			`connect to sample;`
			• `list tablespaces show detail;`
			• `get snapshot for tablespaces on sample;`
			`connect reset;`
			
			Die für USERSPACE1 zurückgegebenen Informationen zeigen an, dass sich dieser Tabellenbereich im Status 'Reorganisation wird ausgeführt' befindet.
			Anmerkung: Tabellenreorganisationsoperationen, bei denen die Datenbank SAMPLE beteiligt ist, werden meist innerhalb eines kurzen Zeitraums ausgeführt und es ist daher möglicherweise schwierig, den Status 'Reorganisation wird ausgeführt' mithilfe dieses Verfahrens zu beobachten.

Restore anstehend	0x100	Tabellenbereiche für eine Datenbank weisen diesen Status nach der Beendigung des ersten Teils einer umgeleiteten Reorganisationsoperation auf (das heißt, bevor der Befehl **SET TABLESPACE CONTAINERS** abgesetzt wird). Bevor der Tabellenbereich verwendet werden kann, muss für den Tabellenbereich (oder für die gesamte Datenbank) ein Restore ausgeführt werden. Sie können erst eine Verbindung zu der Datenbank herstellen, wenn die Restoreoperation erfolgreich durchgeführt wurde. Dann wird auch der Wert des Datenbankkonfigurationsparameters **restore_pending** (der der Information dient) auf den Wert NO gesetzt.	Wenn der erste Teil der umgeleiteten Restoreoperation in 'Speicher kann definiert werden' abgeschlossen wird, weisen alle Tabellenbereiche den Status 'Restore anstehend' auf.

Tabelle 17. Unterstützte Statuswerte für Tabellenbereiche (Forts.)

Status	Hexadezimaler Statuswert	Beschreibung	Beispiele

Restore wird ausgeführt	0x2000	Dies ist ein transienter Status, der nur während einer Restoreoperation wirksam ist.	`update db cfg for sample using logretain recovery;`
			`backup db sample;`
			`backup db sample tablespace (userspace1);`
			
			Die Zeitmarke für dieses Backup-Image lautet:
			
			20040611174124
			
			`restore db sample tablespace (userspace1) online taken at 20040611174124;`
			
			Führen Sie während der Ausführung der Restoreoperation eines der folgenden Scripts in einer anderen Sitzung aus:
			
			`connect to sample;`
			• `list tablespaces show detail;`
			• `get snapshot for tablespaces on sample;`
			`connect reset;`
			
			Die für USERSPACE1 zurückgegebenen Informationen zeigen an, dass sich dieser Tabellenbereich im Status 'Restore wird ausgeführt' befindet.

Aktualisierende Recovery anstehend	0x80	In diesem Status befindet sich ein Tabellenbereich, nachdem eine Restoreoperation für eine wiederherstellbare Datenbank ausgeführt wurde. Bevor der Tabellenbereich verwendet werden kann, muss für den Tabellenbereich (oder für die gesamte Datenbank) ein Restore ausgeführt werden. Eine Datenbank ist wiederherstellbar, wenn für den Datenbankkonfigurationsparameter **logretain** der Wert RECOVERY eingestellt ist oder wenn der Datenbankkonfigurationsparameter **userexit** auf den Wert YES eingestellt ist. Sie können die Datenbank erst aktivieren oder eine Verbindung zu ihr herstellen, wenn eine aktualisierende Recovery erfolgreich durchgeführt wurde. Dann wird auch der Wert des Datenbankkonfigurationsparameters **rollfwd_pending** (der der Information dient) auf den Wert NO gesetzt.	Wenn die Online-Restoreoperation für den Tabellenbereich in 'Restore anstehend' abgeschlossen wird, weist der Tabellenbereich USERSPACE1 den Status 'Aktualisierende Recovery anstehend' auf.

Tabelle 17. Unterstützte Statuswerte für Tabellenbereiche (Forts.)

Status	Hexadezimaler Statuswert	Beschreibung	Beispiele

Aktualisierende Recovery wird ausgeführt	0x40	Dies ist ein transienter Status, der nur während einer aktualisierenden Recovery wirksam ist.	Eingabedatei `staffdata.del` zum Laden mit einem erheblichen Datenvolumen (z. B. 20000 oder mehr Datensätze):
			`update db cfg for sample using logretain recovery;`
			`backup db sample;`
			`connect to sample;`
			`create tablespace ts1 managed by database using (file '/home/melnyk/melnyk/NODE0000/SQL00001 /ts1c1' 1024);`
			`create table newstaff like staff in ts1;`
			`connect reset;`
			`backup db sample tablespace (ts1) online;`
			
			Die Zeitmarke für dieses Backup-Image lautet:
			
			20040630000715
			`connect to sample;`
			`load from staffdata.del of del insert into newstaff copy yes to /home/melnyk/backups;`
			`connect reset;`
			`restore db sample tablespace (ts1) online taken at 20040630000715;`
			`rollforward db sample to end of logs and stop tablespace (ts1) online;`
			
			Führen Sie während der aktualisierenden Recovery eines der folgenden Scripts in einer anderen Sitzung aus:
			`connect to sample;`
			• `list tablespaces show detail;`
			• `get snapshot for tablespaces on sample; connect reset;`
			
			Die für TS1 zurückgegebenen Informationen zeigen an, dass sich dieser Tabellenbereich im Status 'Aktualisierende Recovery wird ausgeführt' befindet.

Speicher kann definiert werden	0x2000000	Tabellenbereiche für eine Datenbank weisen diesen Status nach der Beendigung des ersten Teils einer umgeleiteten Reorganisationsoperation auf (das heißt, bevor der Befehl **SET TABLESPACE CONTAINERS** abgesetzt wird). Dadurch können Sie Container neu definieren.	`backup db sample;`
			
			Unter der Voraussetzung, dass die Zeitmarke für dieses Backup-Image 20040613204955 lautet:
			`restore db sample taken at 20040613204955 redirect;`
			`list tablespaces;`
			
			Die Informationen, die durch den Befehl **LIST TABLESPACES** zurückgegeben werden, zeigen an, dass sich alle Tabellenbereiche im Status 'Speicher kann definiert werden' und 'Restore anstehend' befinden.

Tabelle 17. Unterstützte Statuswerte für Tabellenbereiche (Forts.)

Status	Hexadezimaler Statuswert	Beschreibung	Beispiele

Speicher muss definiert werden	0x1000	Tabellenbereiche für eine Datenbank weisen diesen Status während einer auf eine neue Datenbank umgeleiteten Restoreoperation auf, wenn die Phase zum Festlegen von Tabellenbereichscontainern ausgelassen wird oder wenn während der Phase zum Festlegen von Tabellenbereichscontainern die angegebenen Container nicht angefordert werden können. Das letztere kann auftreten, wenn z. B. ein ungültiger Pfadname angegeben wurde oder wenn nicht genügend Plattenspeicherplatz vorhanden ist.	`backup db sample;`
			
			Unter der Voraussetzung, dass die Zeitmarke für dieses Backup-Image 20040613204955 lautet:
			`restore db sample taken at 20040613204955 into mydb redirect;`
			`set tablespace containers for 2 using (path 'ts2c1');`
			`list tablespaces;`
			
			Die Informationen, die durch den Befehl **LIST TABLESPACES** zurückgegeben werden, zeigen an, dass sich die Tabellenbereiche SYSCATSPACE und TEMPSPACE1 im Status 'Speicher muss definiert werden', 'Speicher kann definiert werden' und 'Restore anstehend' befinden. Der Status 'Speicher muss definiert werden' hat Vorrang vor dem Status 'Speicher kann definiert werden'.

Schreiboperation zurückstellen	0x10000	In diesem Status befindet sich ein Tabellenbereich, nachdem eine Schreiboperation zurückgestellt wurde.	Ein Beispiel für diesen Tabellenbereichsstatus würde den Rahmen dieses Dokuments sprengen.

Tabellenbereich wird erstellt	0x40000000	Dies ist ein transienter Status, der nur während einer Operation zum Erstellen von Tabellenbereichen wirksam ist.	`connect to sample;`
			`create tablespace ts1 managed by database using (file '/home/melnyk/melnyk/NODE0000/SQL00001/tsc1' 1024);`
			`create tablespace ts2 managed by database using (file '/home/melnyk/melnyk/NODE0000/SQL00001/tsc2' 1024);`
			`create tablespace ts3 managed by database using (file '/home/melnyk/melnyk/NODE0000/SQL00001/tsc3' 1024);`
			
			Führen Sie während der Ausführung der Operationen zum Erstellen von Tabellenbereichen eines der folgenden Scripts in einer anderen Sitzung aus:
			`connect to sample;`
			• `list tablespaces show detail;`
			• `get snapshot for tablespaces on sample;`
			`connect reset;`
			
			Die für TS1, TS2 und TS3 zurückgegebenen Informationen zeigen an, dass diese Tabellenbereiche den Status 'Tabellenbereich wird erstellt' aufweisen.

Tabelle 17. Unterstützte Statuswerte für Tabellenbereiche (Forts.)

Status	Hexadezimaler Statuswert	Beschreibung	Beispiele

Tabellenbereich wird gelöscht	0x20000000	Dies ist ein transienter Status, der nur während einer Operation zum Löschen von Tabellenbereichen wirksam ist.	`connect to sample;`
			`create tablespace ts1 managed by database using (file '/home/melnyk/melnyk/NODE0000/SQL00001 /tsc1' 1024);`
			`create tablespace ts2 managed by database using (file '/home/melnyk/melnyk/NODE0000/SQL00001 /tsc2' 1024);`
			`create tablespace ts3 managed by database using (file '/home/melnyk/melnyk/NODE0000/SQL00001 /tsc3' 1024);`
			`drop tablespace ts1;`
			`drop tablespace ts2;`
			`drop tablespace ts3;`
			
			Führen Sie während der Ausführung der Operationen zum Löschen von Tabellenbereichen eines der folgenden Scripts in einer anderen Sitzung aus:
			`connect to sample;`
			• `list tablespaces show detail;`
			• `get snapshot for tablespaces on sample; connect reset;`
			
			Die für TS1, TS2 und TS3 zurückgegebenen Informationen zeigen an, dass diese Tabellenbereiche den Status 'Tabellenbereich wird gelöscht' aufweisen.

Speichergruppe und Tabellenbereich - Datenträgerattribute

Tabellenbereiche mit dynamischem Speicher übernehmen Werte von Datenträgerattributen (einschließlich der Attribute DEVICE READ RATE und DATA TAG) von der Speichergruppe, die standardmäßig von den Tabellenbereichen verwendet wird.

Bei der Erstellung einer Speichergruppe mit der Anweisung CREATE STOGROUP können Sie die folgenden Speichergruppenattribute angeben:

OVERHEAD
Dieses Attribut gibt die E/A-Controllerzeit sowie die Plattensuch- und -latenzzeit in Millisekunden an.

DEVICE READ RATE
Dieses Attribut gibt die Einheitenspezifikation für die Leseübertragungsrate in Megabyte pro Sekunde an. Anhand dieses Werts werden die E/A-Kosten während der Abfrageoptimierung ermittelt. Wenn dieser Wert nicht für alle Speicherpfade identisch ist, sollte dieser Wert der Durchschnittswert für alle Speicherpfade sein, die zu der Speichergruppe gehören.

DATA TAG
Dieses Attribut gibt einen Tag für die Daten in einer bestimmten Speichergruppe an, mit dessen Hilfe der WLM die Verarbeitungspriorität von Datenbankaktivitäten festlegen kann.

Die Standardwerte für die Speichergruppenattribute lauten wie folgt:

Tabelle 18. Standardeinstellungen für Speichergruppenattribute

Attribut	Standardeinstellung
DATA TAG	NONE
DEVICE READ RATE	100 MB/Sek.
OVERHEAD	6,725 ms

Bei der Erstellung eines Tabellenbereichs mit dynamischem Speicher können Sie einen Tag angeben, der die in diesem Tabellenbereich enthaltenen Daten identifiziert. Wenn dieser Tabellenbereich einer Speichergruppe zugeordnet ist, überschreibt das Attribut DATA TAG für den Tabellenbereich alle DATA TAG-Attribute, die für die Speichergruppe definiert sind. Wenn der Benutzer kein Attribut DATA TAG für den Tabellenbereich definiert und der Tabellenbereich in einer Speichergruppe enthalten ist, übernimmt der Tabellenbereich den DATA TAG-Wert von der Speichergruppe. Das Attribut DATA TAG kann für alle regulären und großen Tabellenbereiche mit Ausnahme des Katalogtabellenbereichs definiert werden (SQL0109N). Das Attribut DATA TAG kann für temporäre Tabellenbereiche nicht definiert werden und gibt den Nachrichtenfehler SQL0109N zurück.

Ein Tabellenbereich mit dynamischem Speicher übernimmt die Attribute OVERHEAD und TRANSFERRATE aus der verwendeten Speichergruppe. Wenn eine Tabelle das Attribut TRANSFERRATE von der verwendeten Speichergruppe übernimmt, wird der Wert von DEVICE READ RATE der Speichergruppe unter Berücksichtigung der Einstellung für PAGESIZE des Tabellenbereichs von Millisekunden pro Seitenleseoperation wie folgt konvertiert:

```
TRANSFERRATE = ( 1 / DEVICE READ RATE ) * 1000 / 1024000 * PAGESIZE
```

Die Einstellung für PAGESIZE verfügt bei einem Tabellenbereich sowohl mit als auch ohne dynamischen Speicher über die entsprechenden Standardwerte für TRANSFERRATE:

Tabelle 19. Standardwerte für TRANSFERRATE

PAGESIZE	TRANSFERRATE
4 KB	0,04 Millisekunden pro Seitenleseoperation
8 KB	0,08 Millisekunden pro Seitenleseoperation
16 KB	0,16 Millisekunden pro Seitenleseoperation
32 KB	0,32 Millisekunden pro Seitenleseoperation

Die Datenträgerattribute DATA TAG, DEVICE READ RATE und OVERHEAD für Tabellenbereiche mit dynamischem Speicher können geändert werden, um eine dynamische Übernahme der Werte aus der zugeordneten Speichergruppe zu ermöglichen. Zur dynamischen Aktualisierung der Datenträgerattribute muss für die Anweisung CREATE TABLESPACE oder die Anweisung ALTER TABLESPACE die Option INHERIT angegeben werden.

Wenn ein Tabellenbereich den Wert eines Attributs aus einer Speichergruppe übernimmt, wird in der Sicht der Katalogtabelle SYSCAT.TABLESPACES der Wert -1 für dieses Attribut zurückgemeldet. Sie können die tatsächlichen Werte der Attribute OVERHEAD, TRANSFERRATE und DATA TAG während der Ausführung mit der folgenden Abfrage anzeigen:

```
select tbspace,
       cast(case when a.datatag = -1 then b.datatag else a.datatag end as smallint)
          eff_datatag,
```

```
       cast(case when a.overhead = -1 then b.overhead else a.overhead end as double)
          eff_overhead,
       cast(case when a.transferrate = -1 then (1 / b.devicereadrate) /
          1024 * a.pagesize else a.transferrate end as double) eff_transferrate
 from syscat.tablespaces a left outer join syscat.stogroups b on a.sgid = b.sgid
```

Bei einem Upgrade auf Version 10.1 werden die Einstellungen für OVERHEAD und TRANSFERRATE der vorhandenen Tabellenbereiche beibehalten; die Attribute OVERHEAD und DEVICE READ RATE für die Speichergruppe werden auf nicht definiert gesetzt. Die neu erstellten Tabellenbereiche in einer Speichergruppe, deren Attribut DEVICE READ RATE nicht definiert ist, verwendet die DB2-Datenbankstandardwerte, die bei der ursprünglichen Erstellung der Datenbank definiert wurden. Wenn die Datenträgereinstellungen der Speichergruppe einen gültigen Wert enthalten, übernimmt der neu erstellte Tabellenbereich diese Werte. Mit der Anweisung ALTER STOGROUP können Datenträgerattribute für die Speichergruppe definiert werden. Bei Tabellenbereichen ohne dynamischen Speicher werden die Datenträgerattribute beibehalten.

Umschalten von Tabellenbereichen vom Offlinestatus in den Onlinestatus

Die Klausel SWITCH ONLINE der Anweisung ALTER TABLESPACE kann verwendet werden, um den Status OFFLINE von einem Tabellenbereich zu entfernen, wenn auf die Container, die diesem Tabellenbereich zugeordnet sind, zugegriffen werden kann.

Vorgehensweise

Geben Sie in der Befehlszeile Folgendes ein, um den Status OFFLINE vom Tabellenbereich zu entfernen.

```
db2 ALTER TABLESPACE name
   SWITCH ONLINE
```

Stattdessen können Sie auch die Verbindungen der Anwendungen zu der Datenbank trennen und anschließend erneut über die Anwendungen herstellen lassen. Dadurch wird der Status OFFLINE vom Tabellenbereich entfernt.

Ergebnisse

Der Status OFFLINE wird vom Tabellenbereich entfernt, während der Rest der Datenbank weiterhin aktiv ist und verwendet wird.

Optimieren der Leistung von Tabellenbereichen bei Datenspeicherung auf RAID-Einheiten

Befolgen Sie diese Richtlinien, um die Leistung zu optimieren, wenn Daten auf RAID-Einheiten (Redundant Array of Independent Disks) gespeichert werden.

Vorgehensweise

- Wenn Sie einen Tabellenbereich auf einer Gruppe von RAID-Einheiten erstellen, erstellen Sie die Container für den Tabellenbereich (SMS oder DMS) auf separaten Einheiten.

  Angenommen Sie verfügen über fünfzehn 146-GB-Platten, die als drei RAID-5-Arrays mit fünf Platten in jedem Array konfiguriert sind. Nach der Formatierung kann jede Platte ungefähr 136 GB an Daten enthalten. Jeder Array kann

daher ungefähr 544 GB (4 aktive Platten x 136 GB) speichern. Wenn Sie über einen Tabellenbereich verfügen, der einen Speicher von 300 GB benötigt, erstellen Sie drei Container, und stellen Sie jeden Container auf eine separate Einheit. Jeder Container belegt 100 GB (300 GB/3) auf einer Einheit, und es verbleiben 444 GB (544 GB - 100 GB) auf jeder Einheit für weitere Tabellenbereiche.

- Wählen Sie eine angemessene Speicherbereichsgröße für die Tabellenbereiche aus.

  Die Speicherbereichsgröße für einen Tabellenbereich ist das Datenvolumen, das der Datenbankmanager in einen Container schreibt, bevor er Daten in den nächsten Container schreibt. Idealerweise sollte die Speicherbereichsgröße ein Vielfaches von der zugrunde liegenden Segmentgröße der Platten sein, wobei die Segmentgröße das Datenvolumen ist, das der Plattencontroller auf eine physische Platte schreibt, bevor er Daten auf die nächste physische Platte schreibt. Die Auswahl einer Speicherbereichsgröße, die ein Vielfaches der Segmentgröße bildet, stellt sicher, dass die auf dem Speicherbereich basierenden Operationen, wie z. B. paralleles sequenzielles Lesen beim Vorablesezugriff, nicht um dieselben physischen Platten konkurrieren. Wählen Sie außerdem eine Speicherbereichsgröße aus, die ein Vielfaches der Seitengröße ist.

  Wenn die Segmentgröße im Beispiel 64 KB beträgt und die Seitengröße 16 KB, könnten 256 KB eine angemessene Speicherbereichsgröße ausmachen.

- Verwenden Sie die Registrierdatenbankvariable **DB2_PARALLEL_IO**, um die parallele Ein-/Ausgabe für alle Tabellenbereiche zu aktivieren und um die Anzahl physischer Platten pro Container anzugeben.

  Legen Sie für die Situation im Beispiel Folgendes fest: **DB2_PARALLEL_IO** = `*:4`.

  Wenn Sie die Vorablesezugriffsgröße eines Tabellenbereichs mit AUTOMATIC definieren, verwendet der Datenbankmanager den Wert von der Anzahl physischer Platten, die Sie für **DB2_PARALLEL_IO** angegeben haben, um den Wert für die Vorablesezugriffsgröße festzulegen. Wenn die Vorablesezugriffsgröße nicht mit AUTOMATIC definiert ist, können Sie sie manuell festlegen, berücksichtigen Sie dabei die RAID-Stripegröße, die der Wert der Segmentgröße multipliziert mit der Anzahl aktiver Platten ist. Wählen Sie einen Wert für die Vorablesezugriffsgröße aus, der die folgenden Bedingungen erfüllt:

  – Er entspricht dem Produkt aus der RAID-Stripegröße multipliziert mit der Anzahl paralleler RAID-Einheiten (oder einer ganzzahligen Darstellung dieses Produkts).

  – Er ist eine ganzzahlige Darstellung der Speicherbereichsgröße.

  In dem Beispiel definieren Sie die Vorablesezugriffsgröße möglicherweise auf 768 KB. Dieser Wert entspricht dem Produkt aus der RAID-Stripegröße (256 KB) multipliziert mit der Anzahl paralleler RAID-Einheiten (3). Er ist auch ein Vielfaches der Speicherbereichsgröße (256 KB). Die Auswahl dieser Vorablesezugriffsgröße bedeutet, dass ein einzelner Vorablesezugriff alle Platten in allen Arrays betrifft. Wenn Sie wollen, dass die Vorablesefunktionen intensiver ausgeführt werden, weil Ihre Auslastung hauptsächlich sequenzielle Suchen umfasst, können Sie stattdessen ein Vielfaches dieses Werts verwenden, wie z. B. 1536 KB (768 KB x 2).

- Definieren Sie nicht die Registrierdatenbankvariable **DB2_USE_PAGE_CONTAINER_TAG**. Wie weiter oben beschrieben, sollten Sie einen Tabellenbereich mit einem Wert für EXTENTSIZE erstellen, der der RAID-Stripegröße oder einem Vielfachen dieser Größe entspricht. Wenn **DB2_USE_PAGE_CONTAINER_TAG** jedoch mit dem Wert `ON` definiert wird, wird eine Containerkennung in der Größe einer Seite verwendet und die EXTENTSIZE-Bereiche richten sich nicht nach den einheitenübergreifend gespeicherten RAID-

Datenblöcken (Stripes) aus. Infolgedessen kann bei einer E/A-Anforderung ein Zugriff auf mehr als die optimale Anzahl physischer Platten erforderlich werden.

Löschen von Tabellenbereichen

Beim Löschen eines Tabellenbereichs werden alle Daten in diesem Tabellenbereich gelöscht, die Container freigegeben, die Katalogeinträge entfernt und alle Objekte, die in dem Tabellenbereich definiert sind, entweder gelöscht oder als ungültig markiert.

Informationen zu diesem Vorgang

Die Container in einem leeren Tabellenbereich können erneut verwendet werden, indem der Tabellenbereich gelöscht wird. Allerdings muss die Anweisung DROP TABLESPACE mit COMMIT festgeschrieben werden, bevor Sie versuchen, die Container erneut zu verwenden.

Anmerkung: Sie können einen Tabellenbereich nicht löschen, ohne nicht auch alle Tabellenbereiche zu löschen, die diesem zugeordnet sind. Wenn Sie zum Beispiel eine Tabelle in einem Tabellenbereich haben und der zugehörige Index in einem anderen Tabellenbereich erstellt wurde, müssen Sie sowohl den Indextabellenbereich als auch den Datentabellenbereich über eine einzige Anweisung DROP TABLESPACE löschen.

Vorgehensweise

- Löschen von Benutzertabellenbereichen:

  Sie können einen Benutzertabellenbereich löschen, der sämtliche Tabellendaten einschließlich Index- und LOB-Daten innerhalb dieses einen Benutzertabellenbereichs enthält. Sie können auch einen Benutzertabellenbereich löschen, der möglicherweise Tabellen enthält, die sich über mehrere Tabellenbereiche erstrecken. Dabei können sich die Tabellendaten in einem Tabellenbereich, die Indizes in einem anderen Tabellenbereich und die LOB-Daten in einem dritten Tabellenbereich befinden. Sie müssen alle drei Tabellenbereiche gleichzeitig mit einer einzigen Anweisung löschen. Wenn sich Tabellen über mehrere Tabellenbereiche erstrecken, muss die Löschanweisung alle Tabellenbereiche enthalten, auf die sich diese Tabellen erstrecken, sonst kann sie nicht erfolgreich ausgeführt werden.

  Mit der folgenden SQL-Anweisung wird der Tabellenbereich ACCOUNTING gelöscht:

  ```
  DROP TABLESPACE ACCOUNTING
  ```

- Löschen von Tabellenbereichen für temporäre Benutzertabellen:

  Ein Tabellenbereich für temporäre Benutzertabellen kann nur gelöscht werden, wenn in diesem Tabellenbereich momentan keine deklarierten oder erstellten temporären Tabellen definiert sind. Wenn Sie den Tabellenbereich löschen, wird nicht versucht, alle deklarierten oder erstellten temporären Tabellen dieses Tabellenbereichs zu löschen.

  Anmerkung: Eine deklarierte oder erstellte temporäre Tabelle wird implizit gelöscht, wenn die Anwendung, die zur Deklaration verwendet wurde, die Verbindung zur Datenbank trennt.

- Löschen von Tabellenbereichen für temporäre Systemtabellen:

  Sie können einen Tabellenbereich für temporäre Systemtabellen, der eine Seitengröße von 4 KB hat, nur löschen, wenn Sie zuvor einen anderen Tabellenbereich für temporäre Systemtabellen erstellt haben. Der neue Tabellenbereich für tem-

poräre Systemtabellen muss eine Seitengröße von 4 KB haben, weil die Datenbank immer über mindestens einen Tabellenbereich für temporäre Systemtabellen verfügen muss, der eine Seitengröße von 4 KB hat. Wenn Sie zum Beispiel über einen einzigen Tabellenbereich für temporäre Systemtabellen mit einer Seitengröße von 4 KB verfügen und diesem einen Container hinzufügen möchten und es sich um einen SMS-Tabellenbereich handelt, müssen Sie zunächst einen neuen Tabellenbereich für temporäre Systemtabellen mit einer Seitengröße von 4 KB mit der entsprechenden Anzahl von Containern hinzufügen, und anschließend den alten Tabellenbereich für temporäre Systemtabellen löschen. (Wenn Sie mit DMS-Tabellenbereichen arbeiten, können Sie einen Container hinzufügen, ohne den Tabellenbereich löschen und erneut erstellen zu müssen.)

Die Standardseitengröße für Tabellenbereiche ist die Seitengröße, mit der die Datenbank erstellt wurde (standardmäßig 4 KB, kann jedoch auch 8 KB, 16 KB oder 32 KB sein).

1. Setzen Sie zum Erstellen eines Tabellenbereichs für temporäre Systemtabellen die folgende Anweisung ab:

   ```
   CREATE SYSTEM TEMPORARY TABLESPACE name
     MANAGED BY SYSTEM USING ('verzeichnisse')
   ```

2. Geben Sie anschließend zum Löschen eines Systemtabellenbereichs in die Befehlszeile die folgende Anweisung ein:

   ```
   DROP TABLESPACE name
   ```

3. Mit der folgenden SQL-Anweisung können Sie einen Tabellenbereich für temporäre Systemtabellen mit dem Namen TEMPSPACE2 erstellen:

   ```
   CREATE SYSTEM TEMPORARY TABLESPACE TEMPSPACE2
     MANAGED BY SYSTEM USING ('d:\systemp2')
   ```

4. Nach der Erstellung von TEMPSPACE2 können Sie den ursprünglichen Tabellenbereich für temporäre Systemtabellen TEMPSPACE1 mit der folgenden Anweisung löschen:

   ```
   DROP TABLESPACE TEMPSPACE1
   ```

Kapitel 9. Speichergruppen

Eine Speichergruppe ist eine benannte Gruppe von Speicherpfaden, in denen Daten gespeichert werden können. Speichergruppen werden konfiguriert, um unterschiedliche Speicherklassen darzustellen, die dem Datenbanksystem zur Verfügung stehen. Der Speichergruppe, die für die Daten am besten geeignet ist, können Tabellenbereiche zugeordnet werden. Speichergruppen werden nur von Tabellenbereichen mit dynamischem Speicher verwendet.

Ein Tabellenbereich kann nur einer Speichergruppe zugeordnet werden, aber eine Speichergruppe kann mehrere Tabellenbereichszuordnungen aufweisen. Zur Verwaltung von Speichergruppenobjekten können Sie die Anweisungen CREATE STOGROUP, ALTER STOGROUP, RENAME STOGROUP, DROP und COMMENT verwenden.

Mit der Tabellenpartitionierungsfunktion können Tabellendaten in mehreren Tabellenbereichen abgelegt werden. Unter Verwendung dieser Funktion können Speichergruppen einen Teil der Tabellendaten im schnellen Speicher speichern, während sich die restlichen Daten in einer oder mehreren Speicherebenen des langsameren Speichers befinden. Mit Speichergruppen kann die Datenspeicherung nach Zugriffshäufigkeit (MTS = Multi-Temperature Storage) unterstützt werden, der Daten basierend auf Speicherklassen priorisiert. Beispielsweise können Speichergruppen erstellt werden, die unterschiedlichen Speicherebenen im Datenbanksystem zugeordnet werden. Anschließend werden die definierten Tabellenbereiche diesen Speichergruppen zugeordnet.

Stellen Sie beim Definieren von Speichergruppen sicher, dass die Speicherpfade gemäß den jeweiligen Servicequalitätsmerkmalen gruppiert werden. Die allgemeinen *Servicequalitätsmerkmale* für Daten orientieren sich an einem Alterungsmuster, bei dem ein häufiger Zugriff auf die neuesten Daten erfolgt und deshalb die schnellste Zugriffszeit benötigt wird (*heiße Daten*), während der Zugriff auf ältere Daten weniger häufig erfolgt und die Zugriffszeit langsamer sein kann (*warme Daten* oder *kalte Daten*). Die Priorität der Daten basiert auf den folgenden Faktoren:

- Häufigkeit des Zugriffs
- Zulässige Zugriffszeit
- Flüchtigkeit der Daten
- Anwendungsvoraussetzungen

Normalerweise verhält sich die Datenpriorität proportional umgekehrt zum Volumen, d. h. es gibt deutlich mehr kalte und warme Daten und nur eine kleine Menge heißer Daten. Mit DB2 Workload Manager (WLM) können Sie Regeln zur Verarbeitung von Aktivitäten definieren, und zwar auf der Basis eines Tags, der Daten, auf die zugegriffen wurde, über die Definition eines Tabellenbereichs oder einer Speichergruppe zugewiesen werden kann.

Datenverwaltung mithilfe der Datenspeicherung nach Zugriffshäufigkeit (MTS = Multi-Temperature Storage)

Sie können Ihre Datenbanken so konfigurieren, dass häufig benötigte Daten (*heiße Daten*) im schnellen Speicher, nicht so häufig benötigte Daten (*warme Daten*) im etwas langsameren Speicher und selten benötige Daten (*kalte Daten*) im langsamen, weniger kostenintensiven Speicher gespeichert werden. Wenn heiße Daten abkühlen und weniger häufig benötigt werden, können diese dynamisch in den langsameren Speicher versetzt werden.

In Datenbanksystemen ist es häufig der Fall, dass nur ein relativ kleiner Teil der Daten heiße Daten sind; die Mehrzahl der Daten sind warme oder kalte Daten. Solche Gruppen von *Daten mit unterschiedlichen Zugriffshäufigkeiten* (MTS-Daten, Multi-Temperature Storage) stellen Sie vor beträchtliche Herausforderungen, wenn Sie die Verwendung von schnellen Speichereinheiten optimieren möchten, indem Sie dort keine kalten Daten speichern. Aufgrund der Tatsache, dass Data-Warehouses immer größere Speichermengen verbrauchen, wird die Optimierung des schnellen Speichers bei der Verwaltung der Speicherkosten immer wichtiger.

Speichergruppen sind Gruppen von Speicherpfaden mit ähnlichen Eigenschaften. Einige kritische Attribute des zugrunde liegenden Speichers, die bei der Erstellung oder Änderung einer Speichergruppe zu berücksichtigen sind, sind die verfügbare Speicherkapazität, die Latenz, die Datenübertragungsgeschwindigkeiten und der Grad des RAID-Schutzes. Es können Speichergruppen erstellt werden, die im Datenbankverwaltungssystem unterschiedlichen Speicherklassen zugeordnet werden. Diesen Speichergruppen können Tabellenbereiche mit dynamischem Speicher zugeordnet werden, und zwar im Hinblick darauf, welche Tabellenbereiche heiße, warme oder kalte Daten enthalten. Um DMS-Tabellenbereiche für die Verwendung von dynamischem Speicher umzuwandeln, müssen Sie eine Anweisung ALTER TABLESPACE mit der Option MANAGED BY AUTOMATIC STORAGE absetzen und anschließend eine Neuverteilungsoperation ausführen.

Aktuelle Daten werden häufig als heiße Daten betrachtet, werden mit zunehmendem Alter aber in der Regel zu warmen oder kalten Daten. Verwenden Sie die Anweisung ALTER TABLESPACE mit der Option USING STOGROUP, um eine dynamische Neuzuordnung eines Tabellenbereichs zu einer anderen Speichergruppe vorzunehmen.

Das folgende Beispiel veranschaulicht die Verwendung von Speichergruppen mit Daten mit unterschiedlicher Zugriffshäufigkeit. Nehmen wir an, dass Sie der Datenbankadministrator eines Unternehmens sind, in dem der Großteil der Verarbeitung für die Daten des aktuellen Finanzquartals erfolgt. Wie in Abb. 32 auf Seite 285 gezeigt verfügt dieses Unternehmen über ausreichende SSD-Kapazität (SSD - Solid-State Drive, Solid-State-Laufwerk) für die Daten eines ganzen Quartals sowie über genügend FC-basierte und SAS-Laufwerkkapazität (FC - Fibre Channel, SAS -Serial-Attached SCSI) für die Daten des restlichen Jahres. Daten, die älter als ein Jahr sind, werden in einem großen SATA-RAID-Array (SATA - Serial Advanced Technology Attachment) gespeichert, das zwar stabil, aber nicht leistungsfähig genug ist, um einer hohen Abfrageauslastung Stand zu halten. Es können drei Speichergruppen definiert werden: eine für den SSD-Speicher (`sg_hot`), eine für den FC- und SAS-Speicher (`sg_warm`) und eine für den SATA-Speicher (`sg_cold`). Führen Sie dann die folgenden Aktionen aus:

- Ordnen Sie den Tabellenbereich mit den Daten des aktuellen Quartals der Speichergruppe 'sg_hot' zu.

- Ordnen Sie den Tabellenbereich mit den Daten der letzten drei Quartale der Speichergruppe 'sg_warm' zu.
- Ordnen Sie den Tabellenbereich mit allen älteren Daten der Speichergruppe 'sg_cold' zu.

Nach Ablauf des aktuellen Quartals führen Sie die folgenden Aktionen aus:

- Ordnen Sie der Speichergruppe 'sg_hot' einen Tabellenbereich für das neue Quartal zu.
- Versetzen Sie den Tabellenbereich für das gerade abgelaufene Quartal in die Speichergruppe 'sg_warm'.
- Versetzen Sie die Daten für das älteste Quartal in der Speichergruppe 'sg_warm' in die Speichergruppe 'sg_cold'.

Diese Arbeiten können ausgeführt werden, während sich die Datenbank im Onlinestatus befindet.

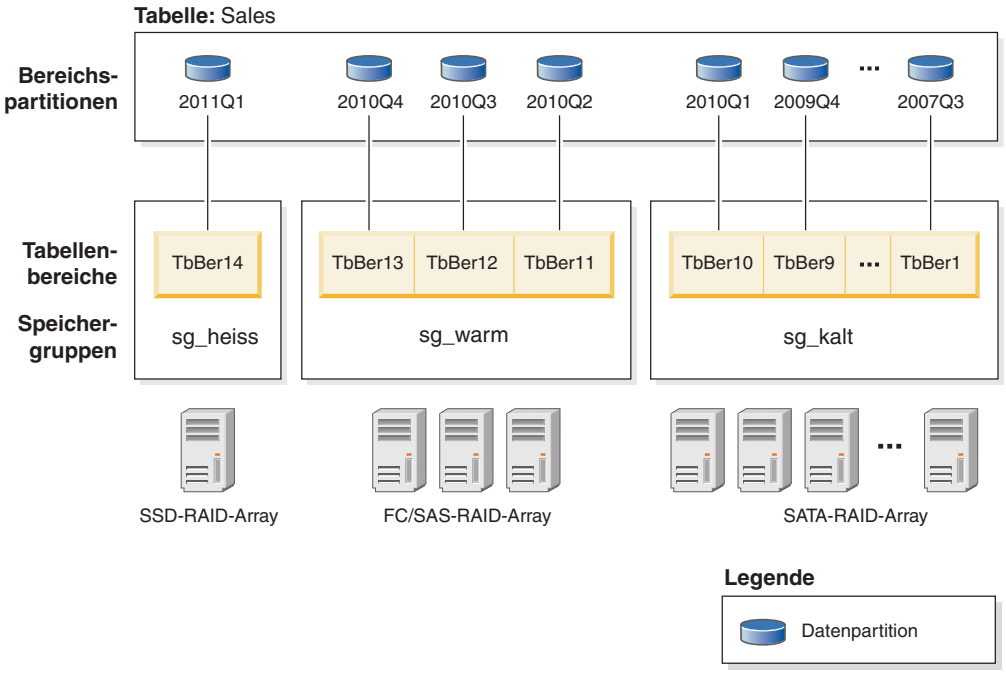


Abbildung 32. Verwalten von Vertriebsdaten mithilfe der Datenspeicherung nach Zugriffshäufigkeit (MTS = Multi-Temperature Storage)

Die folgenden Schritte enthalten weitere Informationen zum Einrichten der Datenspeicherung nach Zugriffshäufigkeit (MTS = Multi-Temperature Storage) für die Vertriebsdaten des aktuellen Finanzjahres.

1. Erstellen Sie zwei Speichergruppen, um die beiden Speicherklassen darzustellen: eine Speichergruppe zur Speicherung von heißen Daten und eine Speichergruppe zur Speicherung von warmen Daten.

   ```
   CREATE STOGROUP sg_hot ON '/ssd/pfad1', '/ssd/path2' DEVICE READ RATE 100 OVERHEAD 6.725
   CREATE STOGROUP sg_warm ON '/hdd/pfad1', '/hdd/pfad2'
   ```

   Diese Anweisungen definieren eine SSD-Speichergruppe (sg_hot) zur Speicherung von heißen Daten sowie eine FC- und SAS-Speichergruppe (sg_warm) zur Speicherung von warmen Daten.
2. Erstellen Sie vier Tabellenbereiche, einen pro Quartal eines Finanzjahres, und ordnen Sie die Tabellenbereiche den Speichergruppen zu.

```
CREATE TABLESPACE tbsp_2010q2 USING STOGROUP sg_warm
CREATE TABLESPACE tbsp_2010q3 USING STOGROUP sg_warm
CREATE TABLESPACE tbsp_2010q4 USING STOGROUP sg_warm
CREATE TABLESPACE tbsp_2011q1 USING STOGROUP sg_hot
```

Diese Zuordnung führt dazu, dass die Tabellenbereiche die Eigenschaften der Speichergruppen übernehmen.

3. Definieren Sie die Bereichspartitionen in der Tabelle SALES.

```
CREATE TABLE sales (order_date DATE, order_id INT, cust_id BIGINT)
PARTITION BY RANGE (order_date)
(PART "2010Q2" STARTING ('2010-04-01') ENDING ('2010-06-30') in "tbsp_2010q2",
 PART "2010Q3" STARTING ('2010-07-01') ENDING ('2010-09-30') in "tbsp_2010q3",
 PART "2010Q4" STARTING ('2010-10-01') ENDING ('2010-12-31') in "tbsp_2010q4",
 PART "2011Q1" STARTING ('2011-01-01') ENDING ('2011-03-31') in "tbsp_2011q1");
```

Die '2011Q1'-Daten stellen die Daten des aktuellen Finanzquartals dar und verwenden die Speichergruppe `sg_hot`.

4. Nach Ablauf des aktuellen Quartals erstellen Sie einen Tabellenbereich für ein neues Quartal und ordnen den Tabellenbereich der Speichergruppe 'sg_hot' zu.

```
CREATE TABLESPACE tbsp_2011q2 USING STOGROUP sg_hot
```

5. Versetzen Sie den Tabellenbereich für das gerade abgelaufene Quartal in die Speichergruppe 'sg_warm'. Zum Ändern der Speichergruppenzuordnung für den Tabellenbereich 'tbsp_2011q1' müssen Sie die Anweisung ALTER TABLESPACE mit der Option USING STOGROUP eingeben.

```
ALTER TABLESPACE tbsp_2011q1 USING STOGROUP sg_warm
```

Standardspeichergruppen

Wenn eine Datenbank über Speichergruppen verfügt, wird bei der Erstellung eines über den dynamischen Speicher verwalteten Tabellenbereichs die Standardspeichergruppe verwendet, ohne die Speichergruppe explizit anzugeben.

Bei der Erstellung einer Datenbank wird automatisch eine Standardspeichergruppe mit dem Namen IBMSTOGROUP erstellt. Eine mit der Klausel AUTOMATIC STORAGE NO erstellte Datenbank verfügt jedoch über keine Standardspeichergruppe. Die erste mit der Anweisung CREATE STOGROUP erstellte Speichergruppe wird als Standardspeichergruppe festgelegt. Es kann nur eine Speichergruppe als Standardspeichergruppe festgelegt werden.

Anmerkung: Eine Datenbank kann mit der Klausel AUTOMATIC STORAGE NO erstellt werden; die Klausel AUTOMATIC STORAGE ist jedoch veraltet und wird in zukünftigen Releases möglicherweise nicht mehr enthalten sein.

Eine Standardspeichergruppe kann mit der Anweisung CREATE STOGROUP oder mit der Anweisung ALTER STOGROUP festgelegt werden. Wenn Sie eine andere Speichergruppe als Standardspeichergruppe festlegen, hat dies keine Auswirkungen auf die vorhandenen Tabellenbereiche, die mit der alten Standardspeichergruppe arbeiten. Verwenden Sie zum Ändern der einem Tabellenbereich zugeordneten Speichergruppe die Anweisung ALTER TABLESPACE.

Über die Katalogsicht SYSCAT.STOGROUPS kann festgestellt werden, welche Speichergruppe die Standardspeichergruppe ist.

Die aktuelle Standardspeichergruppe kann nicht gelöscht werden. Die Speichergruppe IBMSTOGROUP kann gelöscht werden, wenn sie zum gegebenen Zeit-

punkt nicht als Standardspeichergruppe definiert ist. Wenn Sie die Speichergruppe IBMSTOGROUP löschen, können Sie eine andere Speichergruppe mit demselben Namen erstellen.

Erstellen von Speichergruppen

Verwenden Sie die Anweisung CREATE STOGROUP, um Speichergruppen zu erstellen. Durch das Erstellen einer Speichergruppe in einer Datenbank werden dieser Speichergruppe Speicherpfade zugeordnet.

Vorbereitende Schritte

Wenn Sie eine Datenbank mit der Klausel `AUTOMATIC STORAGE NO` erstellen, besitzt diese keine Standardspeichergruppe. Mithilfe der Anweisung CREATE STOGROUP können Sie eine Standardspeichergruppe erstellen.

Anmerkung: Eine Datenbank kann mit der Klausel AUTOMATIC STORAGE NO erstellt werden; die Klausel AUTOMATIC STORAGE ist jedoch veraltet und wird in zukünftigen Releases möglicherweise nicht mehr enthalten sein.

Vorgehensweise

Geben Sie die folgende Anweisung in die Befehlszeile ein, um eine Speichergruppe zu erstellen:

```
CREATE STOGROUP operative_speichergruppe ON '/dateisystem1',
  '/dateisystem2', '/dateisystem3'...
```

Dabei ist *operative_speichergruppe* der Name der Speichergruppe und */dateisystem1*, */dateisystem2*, */dateisystem3*, ... steht für die hinzuzufügenden Speicherpfade.

Wichtig: Damit eine vorhersehbare Leistung sichergestellt werden kann, sollten alle einer Speichergruppe zugeordneten Pfade dieselben Datenträgermerkmale aufweisen: Latenzzeit, Einheitenleserate und Größe.

Ändern von Speichergruppen

Mit der Anweisung ALTER STOGROUP kann die Definition einer Speichergruppe geändert werden. Dazu gehört unter anderem das Konfigurieren von Datenträgerattributen, das Festlegen eines Datentags oder das Definieren einer Standardspeichergruppe. Ferner können für eine Speichergruppe Speicherpfade hinzugefügt oder entfernt werden.

Wenn Sie einer Speichergruppe Speicherpfade hinzufügen möchten und die Speicherbereiche der zugehörigen Tabellenbereiche einheitenübergreifend über alle Speicherpfade hinweg verteilt werden sollen, müssen Sie die Anweisung ALTER TABLESPACE mit der Option REBALANCE für jeden Tabellenbereich verwenden, der dieser Speichergruppe zugeordnet ist.

Wenn Sie Speicherpfade aus einer Speichergruppe löschen, müssen Sie die Anweisung ALTER TABLESPACE mit der Option REBALANCE verwenden, um zugeordnete Speicherbereiche aus den gelöschten Pfaden an andere Positionen zu versetzen.

Mit DB2 Workload Manager (WLM) können Sie Regeln zur Verarbeitung von Aktivitäten definieren, und zwar auf der Basis eines Tags, der Daten zugeordnet wird,

auf die zugegriffen wurde. Der Tag wird den Daten beim Definieren eines Tabellenbereichs oder einer Speichergruppe zugeordnet.

Hinzufügen von Speicherpfaden

Mit der Anweisung ALTER STOGROUP kann einer Speichergruppe ein Speicherpfad hinzugefügt werden.

Informationen zu diesem Vorgang

Wenn Sie einen Speicherpfad für eine Umgebung mit mehreren Datenbankpartitionen hinzufügen, muss der Speicherpfad in jeder Datenbankpartition vorhanden sein. Wenn der angegebene Pfad nicht in jeder Datenbankpartition vorhanden ist, wird die Anweisung rückgängig gemacht.

Vorgehensweise

- Geben Sie die folgende Anweisung ALTER STOGROUP ein, um einer Speichergruppe Speicherpfade hinzuzufügen:

  ```
  ALTER STOGROUP sg ADD '/hdd/pfad1', '/hdd/pfad2', ...
  ```

  Dabei ist *sg* die Speichergruppe und */hdd/pfad1*, */hdd/pfad2*, ... sind die hinzugefügten Speicherpfade.

  Wichtig: Alle einer Speichergruppe zugeordneten Pfade sollten ähnliche Datenträgermerkmale aufweisen: zugrunde liegende Platten, Latenzzeit, Einheitenleserate und Größe. Wenn Pfade verschiedene Datenträgermerkmale aufeisen, können die Leistungswerte inkonsistent sein.
- Nach dem Hinzufügen eines oder mehrerer Speicherpfade zur Speichergruppe können Sie optional die Anweisung ALTER TABLESPACE ausführen, um die Daten der Tabellenbereiche neu zu verteilen, sodass unverzüglich mit der Verwendung der neuen Speicherpfade begonnen wird. Ansonsten werden die neuen Speicherpfade nur dann verwendet, wenn in den Containern der vorhandenen Speicherpfade kein Platz mehr verfügbar ist. Führen Sie die folgende Anweisung aus, um alle betroffenen permanenten Tabellenbereiche in der Speichergruppe zu ermitteln:

  ```
  SELECT TBSP_NAME
     FROM table (MON_GET_TABLESPACE(' ', -2))
     WHERE TBSP_USING_AUTO_STORAGE = 1
        AND TBSP_CONTENT_TYPE IN ('ANY','LARGE')
        AND STORAGE_GROUP_NAME = 'sg'
     ORDER BY TBSP_ID
  ```

  Wenn die Tabellenbereiche ermittelt wurden, können Sie die folgende Anweisung für jeden der aufgeführten Tabellenbereiche ausführen:

  ```
  ALTER TABLESPACE tabellenbereichsname REBALANCE
  ```

  Dabei gibt *tabellenbereichsname* den Namen des Tabellenbereichs an.

Löschen von Speicherpfaden

Sie können einen oder mehrere Speicherpfade aus einer Speichergruppe löschen oder Daten aus den Speicherpfaden versetzen und sie neu auf die Speicherpfade verteilen (Neuausgleich).

Vorbereitende Schritte

Über die Verwaltungssicht ADMIN_GET_STORAGE_PATHS können Sie feststellen, ob der Speicherpfad von permanenten Tabellenbereichen verwendet wird. In dieser

Sicht werden aktuelle Informationen zu den Speicherpfaden für jede Speichergruppe angezeigt. Ein Speicherpfad kann einen von drei Status haben:

NOT_IN_USE
Der Speicherpfad wurde der Datenbank hinzugefügt, wird jedoch von keinem Tabellenbereich verwendet.

IN_USE
Mindestens ein Tabellenbereich hat Container im betreffenden Speicherpfad.

DROP_PENDING
Eine Anforderung ALTER STOGROUP *speichergruppenname* DROP wurde abgesetzt, um den Pfad zu löschen, jedoch wird der Speicherpfad noch von Tabellenbereichen verwendet. Der Pfad wird aus der Datenbank entfernt, wenn er von keinem Tabellenbereich mehr verwendet wird.

Wenn in dem von Ihnen gelöschten Speicherpfad Daten gespeichert sind und der Pfad den Status DROP_PENDING (Löschen anstehend) hat, müssen Sie mithilfe des Speicherpfads eine Neuverteilung aller permanenten Tabellenbereiche ausführen, damit der Datenbankmanager die Löschung des Pfads abschließen kann.

Über die Verwaltungssicht MON_GET_TABLESPACE erhalten Sie Informationen zu Tabellenbereichen auf bestimmten Datenbankpartitionen.

Einschränkungen

Eine Speichergruppe muss mindestens einen Pfad aufweisen. Es können nicht alle Pfade in einer Speichergruppe gelöscht werden.

Informationen zu diesem Vorgang

Wenn Sie einen Speicherpfad löschen wollen, müssen Sie alle Tabellenbereiche mit permanenten Daten neu verteilen, die den Speicherpfad verwenden, indem Sie die Anweisung ALTER TABLESPACE tabellenbereichsname REBALANCE ausführen. Dadurch werden die Daten aus dem zu löschenden Pfad entfernt. In diesem Fall versetzt die Umverteilungsoperation Daten aus dem Speicherpfad, der gelöscht werden soll, in die verbleibenden Speicherpfade und verteilt sie einheitenübergreifend und konsistent auf diese Speicherpfade, sodass eine maximale E/A-Parallelität sichergestellt wird.

Vorgehensweise

1. Geben Sie die folgende Anweisung ALTER STOGROUP ein, um Speicherpfade aus einer Speichergruppe zu löschen:
   ```
   ALTER STOGROUP sg DROP '/db2/dateisystem1', '/db2/dateisystem2'
   ```
   Dabei ist *sg* die Speichergruppe und */db2/dateisystem1* und */db2/dateisystem2* sind die Speicherpfade, die gelöscht werden.
2. Führen Sie eine Neuverteilung für die Container der Speicherpfade aus, die gerade gelöscht werden. Geben Sie die folgende Anweisung ein, um alle betroffenen permanenten Tabellenbereiche in der Datenbank zu ermitteln, die Container in einem Pfad mit dem Status 'Löschen anstehend' haben:

```
SELECT TBSP_NAME
   FROM table (MON_GET_TABLESPACE(' ', -2))
   WHERE TBSP_USING_AUTO_STORAGE = 1
      AND TBSP_CONTENT_TYPE IN ('ANY','LARGE')
      AND STORAGE_GROUP_NAME = 'sg'
   ORDER BY TBSP_ID
```

Nachdem die Tabellenbereiche ermittelt wurden, können Sie die folgende Anweisung für jeden der aufgeführten Tabellenbereiche ausführen:

```
ALTER TABLESPACE tabellenbereichsname REBALANCE
```

Dabei gibt *tabellenbereichsname* den Namen eines Tabellenbereichs an.

Nach Abschluss der letzten Neuverteilungsoperation werden die Pfade /db2/dateisystem1 und /db2/dateisystem2 aus der Speichergruppe entfernt.

3. Löschen Sie die Tabellenbereiche für temporäre Tabellen, die die Speichergruppe verwenden. Ein Tabellenbereich im Status DROP_PENDING (Löschen anstehend) wird nicht gelöscht, wenn er einen Tabellenbereich für temporäre Tabellen enthält.
4. Erstellen Sie die Tabellenbereiche für temporäre Tabellen erneut, die die Speichergruppe verwendet haben.

Nächste Schritte

Führen Sie eine Abfrage für die Verwaltungssicht ADMIN_GET_STORAGE_PATHS aus, um sicherzustellen, dass der gelöschte Speicherpfad nicht mehr aufgelistet wird. Er wird aufgelistet, wenn er noch von mindestens einem Tabellenbereich verwendet wird.

Überwachen der Speicherpfade

Mithilfe von Verwaltungssichten und Tabellenfunktionen können Informationen zu den verwendeten Speicherpfaden abgerufen werden.

Die folgenden Verwaltungssichten und Tabellenfunktionen können verwendet werden:

- Über die Verwaltungssicht ADMIN_GET_STORAGE_PATHS können Sie eine Liste der Speicherpfade für jede Speichergruppe sowie Dateisysteminformationen für jeden Speicherpfad abrufen.
- Über die Monitorelemente TBSP_USING_AUTOMATIC_STORAGE und STORAGE_GROUP_NAME in der Tabellenfunktion MON_GET_TABLESPACE kann festgestellt werden, ob ein Tabellenbereich dynamischen Speicher verwendet und welche Speichergruppe der Tabellenbereich verwendet.
- Über das Monitorelement DB_STORAGE_PATH_ID in der Tabellenfunktion MON_GET_CONTAINER kann festgestellt werden, für welchen Speicherpfad in einer Speichergruppe ein Container definiert ist.

Ersetzen der Pfade einer Speichergruppe

Ersetzen Sie die Speicherpfade in einer Speichergruppe durch neue Speicherpfade.

Vorgehensweise

Gehen Sie wie folgt vor, um die vorhandenen Speicherpfade in einer Speichergruppe zu ersetzen:

1. Fügen Sie die neuen Speicherpfade zu einer vorhandenen Speichergruppe hinzu.

```
ALTER STOGROUP sg_default ADD '/hdd/path3', '/hdd/path4'
```

2. Löschen Sie die alten Speicherpfade.

```
ALTER STOGROUP sg_default DROP '/hdd/path1', '/hdd/path2'
```

Anmerkung: Alle Speichergruppen müssen mindestens einen Pfad aufweisen und dieser letzte Pfad kann nicht gelöscht werden.
Dadurch werden die gelöschten Speicherpfade als DROP PENDING markiert.

3. Ermitteln Sie die betroffenen Tabellenbereiche für nicht temporäre Tabellen.

```
SELECT TBSP_NAME
   FROM table (MON_GET_TABLESPACE(' ', -2))
   WHERE TBSP_USING_AUTO_STORAGE = 1
      AND TBSP_CONTENT_TYPE IN ('ANY','LARGE')
      AND STORAGE_GROUP_NAME = 'sg_default'
   ORDER BY TBSP_ID
```

4. Führen Sie für alle betroffenen Tabellenbereiche für nicht temporäre Tabellen, die zurückgegeben werden, die folgende Anweisung aus.

```
ALTER TABLESPACE tabellenbereichsname REBALANCE
```

5. Wenn in den gelöschten Speicherpfaden Tabellenbereiche für temporäre Tabellen definiert wurden, müssen Sie zuerst die neuen Tabellenbereiche für temporäre Tabellen erstellen, bevor Sie die alten löschen.

```
SELECT TBSP_NAME
   FROM table (MON_GET_TABLESPACE(' ', -2))
   WHERE TBSP_USING_AUTO_STORAGE = 1
      AND TBSP_CONTENT_TYPE IN ('USRTEMP','SYSTEMP')
      AND STORAGE_GROUP_NAME = 'sg_default'
   ORDER BY TBSP_ID
```

Umbenennen von Speichergruppen

Zum Umbenennen einer Speichergruppe verwenden Sie die Anweisung RENAME STOGROUP.

Vorgehensweise

Verwenden Sie zum Umbenennen einer Speichergruppe die folgende Anweisung:

```
RENAME STOGROUP sg_hot TO sg_warm
```

Dabei ist `sg_warm` der neue Name der Speichergruppe.

Beispiel

Wenn während der Datenbankerstellung die erste Speichergruppe erstellt wird, lautet der Standardname der Speichergruppe *IBMSTOGROUP*. Sie können den angegebenen Standardnamen mit der folgenden Anweisung ändern:

```
RENAME STOGROUP IBMSTOGROUP TO DEFAULT_SG
```

Dabei ist `DEFAULT_SG` der neue Standardname der Speichergruppe.

Löschen von Speichergruppen

Sie können eine Speichergruppe mit der Anweisung DROP löschen.

Informationen zu diesem Vorgang

Bevor Sie eine Speichergruppe löschen, müssen Sie ermitteln, ob es Tabellenbereiche gibt, die diese Speichergruppe verwenden. Ist dies der Fall, müssen Sie die von

den Tabellenbereichen verwendete Speichergruppe ändern und vor dem Löschen der ursprünglichen Speichergruppe die Neuverteilungsoperation ausführen.

Einschränkungen

Die aktuelle Standardspeichergruppe kann nicht gelöscht werden.

Vorgehensweise

Gehen Sie wie folgt vor, um eine Speichergruppe zu löschen:

1. Suchen Sie die Tabellenbereiche, die die Speichergruppe verwenden.

   ```
   SELECT TBSP_NAME, TBSP_CONTENT_TYPE
      FROM tabelle (MON_GET_TABLESPACE(' ', -2))
      WHERE TBSP_USING_AUTO_STORAGE = 1
         AND STORAGE_GROUP_NAME = speichergruppe
      ORDER BY TBSP_ID
   ```

   Dabei ist *speichergruppe* die Speichergruppe, die Sie löschen wollen.
2. Wenn reguläre oder große Tabellenbereiche die Speichergruppe verwenden, ordnen Sie sie einer anderen Speichergruppe zu:

   ```
   ALTER TABLESPACE tabellenbereichsname USING STOGROUP neue_speichergruppe
   ```

   Dabei ist *neue_speichergruppe* eine andere Speichergruppe.
3. Wenn Tabellenbereiche für temporäre Tabellen die Speichergruppe verwenden, die Sie löschen wollen, führen Sie folgende Schritte aus:
   a. Ermitteln Sie, welche Tabellenbereiche für temporäre Tabellen die zu löschende Speichergruppe verwenden:

      ```
      SELECT TBSP_NAME
         FROM table (MON_GET_TABLESPACE(' ', -2))
         WHERE TBSP_USING_AUTO_STORAGE = 1
            AND TBSP_CONTENT_TYPE IN ('USRTEMP','SYSTEMP')
            AND STORAGE_GROUP_NAME = 'STO_GROUP'
         ORDER BY TBSP_ID
      ```

   b. Löschen Sie die Tabellenbereiche für temporäre Tabellen, die die Speichergruppe verwenden:

      ```
      DROP TABLESPACE tabellenbereich
      ```

   c. Erstellen Sie die Tabellenbereiche für temporäre Tabellen erneut, die die Speichergruppe verwendet haben.
4. Überwachen Sie den Neuverteilungsvorgang für die zu löschende Speichergruppe.

   ```
   SELECT * from table (MON_GET_REBALANCE_STATUS( ' ', -2))
     WHERE REBALANCER_SOURCE_STORAGE_GROUP_NAME = neue_speichergruppe
   ```

   Ein leerer Ergebnisstatus zeigt an, dass alle Tabellenbereiche in die neue Speichergruppe versetzt wurden.
5. Löschen Sie die Speichergruppe, wenn alle EXTENTSIZE großen Speicherbereiche erfolgreich in die Zielspeichergruppe versetzt wurden.

   ```
   DROP STOGROUP speichergruppe
   ```

   Dabei ist *speichergruppe* der Name der zu löschenden Speichergruppe.

Speichergruppe und Tabellenbereich - Datenträgerattribute

Tabellenbereiche mit dynamischem Speicher übernehmen Werte von Datenträgerattributen (einschließlich der Attribute DEVICE READ RATE und DATA TAG) von der Speichergruppe, die standardmäßig von den Tabellenbereichen verwendet wird.

Bei der Erstellung einer Speichergruppe mit der Anweisung CREATE STOGROUP können Sie die folgenden Speichergruppenattribute angeben:

OVERHEAD
Dieses Attribut gibt die E/A-Controllerzeit sowie die Plattensuch- und -latenzzeit in Millisekunden an.

DEVICE READ RATE
Dieses Attribut gibt die Einheitenspezifikation für die Leseübertragungsrate in Megabyte pro Sekunde an. Anhand dieses Werts werden die E/A-Kosten während der Abfrageoptimierung ermittelt. Wenn dieser Wert nicht für alle Speicherpfade identisch ist, sollte dieser Wert der Durchschnittswert für alle Speicherpfade sein, die zu der Speichergruppe gehören.

DATA TAG
Dieses Attribut gibt einen Tag für die Daten in einer bestimmten Speichergruppe an, mit dessen Hilfe der WLM die Verarbeitungspriorität von Datenbankaktivitäten festlegen kann.

Die Standardwerte für die Speichergruppenattribute lauten wie folgt:

Tabelle 20. Standardeinstellungen für Speichergruppenattribute

Attribut	Standardeinstellung
DATA TAG	NONE
DEVICE READ RATE	100 MB/Sek.
OVERHEAD	6,725 ms

Bei der Erstellung eines Tabellenbereichs mit dynamischem Speicher können Sie einen Tag angeben, der die in diesem Tabellenbereich enthaltenen Daten identifiziert. Wenn dieser Tabellenbereich einer Speichergruppe zugeordnet ist, überschreibt das Attribut DATA TAG für den Tabellenbereich alle DATA TAG-Attribute, die für die Speichergruppe definiert sind. Wenn der Benutzer kein Attribut DATA TAG für den Tabellenbereich definiert und der Tabellenbereich in einer Speichergruppe enthalten ist, übernimmt der Tabellenbereich den DATA TAG-Wert von der Speichergruppe. Das Attribut DATA TAG kann für alle regulären und großen Tabellenbereiche mit Ausnahme des Katalogtabellenbereichs definiert werden (SQL0109N). Das Attribut DATA TAG kann für temporäre Tabellenbereiche nicht definiert werden und gibt den Nachrichtenfehler SQL0109N zurück.

Ein Tabellenbereich mit dynamischem Speicher übernimmt die Attribute OVERHEAD und TRANSFERRATE aus der verwendeten Speichergruppe. Wenn eine Tabelle das Attribut TRANSFERRATE von der verwendeten Speichergruppe übernimmt, wird der Wert von DEVICE READ RATE der Speichergruppe unter Berücksichtigung der Einstellung für PAGESIZE des Tabellenbereichs von Millisekunden pro Seitenleseoperation wie folgt konvertiert:

```
TRANSFERRATE = ( 1 / DEVICE READ RATE ) * 1000 / 1024000 * PAGESIZE
```

Die Einstellung für PAGESIZE verfügt bei einem Tabellenbereich sowohl mit als auch ohne dynamischen Speicher über die entsprechenden Standardwerte für TRANSFERRATE:

Tabelle 21. Standardwerte für TRANSFERRATE

PAGESIZE	TRANSFERRATE
4 KB	0,04 Millisekunden pro Seitenleseoperation
8 KB	0,08 Millisekunden pro Seitenleseoperation

Tabelle 21. Standardwerte für TRANSFERRATE (Forts.)

PAGESIZE	TRANSFERRATE
16 KB	0,16 Millisekunden pro Seitenleseoperation
32 KB	0,32 Millisekunden pro Seitenleseoperation

Die Datenträgerattribute DATA TAG, DEVICE READ RATE und OVERHEAD für Tabellenbereiche mit dynamischem Speicher können geändert werden, um eine dynamische Übernahme der Werte aus der zugeordneten Speichergruppe zu ermöglichen. Zur dynamischen Aktualisierung der Datenträgerattribute muss für die Anweisung CREATE TABLESPACE oder die Anweisung ALTER TABLESPACE die Option INHERIT angegeben werden.

Wenn ein Tabellenbereich den Wert eines Attributs aus einer Speichergruppe übernimmt, wird in der Sicht der Katalogtabelle SYSCAT.TABLESPACES der Wert -1 für dieses Attribut zurückgemeldet. Sie können die tatsächlichen Werte der Attribute OVERHEAD, TRANSFERRATE und DATA TAG während der Ausführung mit der folgenden Abfrage anzeigen:

```
 select tbspace,
 cast(case when a.datatag = -1 then b.datatag else a.datatag end as smallint) eff_datatag,
 cast(case when a.overhead = -1 then b.overhead else a.overhead end as double) eff_overhead,
 cast(case when a.transferrate = -1 then (1 / b.devicereadrate) /
 1024 * a.pagesize else a.transferrate end as double) eff_transferrate
from syscat.tablespaces a left outer join syscat.stogroups b on a.sgid = b.sgid
```

Bei einem Upgrade auf Version 10.1 werden die Einstellungen für OVERHEAD und TRANSFERRATE der vorhandenen Tabellenbereiche beibehalten; die Attribute OVERHEAD und DEVICE READ RATE für die Speichergruppe werden auf nicht definiert gesetzt. Die neu erstellten Tabellenbereiche in einer Speichergruppe, deren Attribut DEVICE READ RATE nicht definiert ist, verwendet die DB2-Datenbankstandardwerte, die bei der ursprünglichen Erstellung der Datenbank definiert wurden. Wenn die Datenträgereinstellungen der Speichergruppe einen gültigen Wert enthalten, übernimmt der neu erstellte Tabellenbereich diese Werte. Mit der Anweisung ALTER STOGROUP können Datenträgerattribute für die Speichergruppe definiert werden. Bei Tabellenbereichen ohne dynamischen Speicher werden die Datenträgerattribute beibehalten.

Zuordnen eines Tabellenbereichs zu einer Speichergruppe

Mit der Anweisung CREATE TABLESPACE oder der Anweisung ALTER TABLESPACE können Sie die Speichergruppe angeben oder ändern, die von einem Tabellenbereich verwendet wird. Wenn bei der Erstellung eines Tabellenbereichs keine Speichergruppe angegeben wird, wird die Standardspeichergruppe verwendet.

Informationen zu diesem Vorgang

Wenn Sie die von einem Tabellenbereich verwendete Speichergruppe ändern, wird beim Festschreiben der Anweisung ALTER TABLESPACE eine implizite REBALANCE-Operation ausgeführt. Diese versetzt die Daten aus der Quellenspeichergruppe in die Zielspeichergruppe.

Bei Verwendung von IBM DB2 pureScale Feature wird REBALANCE nicht unterstützt und die zugeordnete Speichergruppe kann nicht geändert werden. Die REBALANCE-Operation ist asynchron und hat keine Auswirkungen auf die Daten-

verfügbarkeit. Mit der Monitortabellenfunktionen MON_GET_REBALANCE_STATUS kann der Fortschritt der REBALANCE-Operation überwacht werden.

Während der Operation ALTER TABLESPACE werden kompilierte Objekte, die auf alten Tabellenbereichsattributen basieren, *vorläufig inaktiviert*. Neue Kompilierungen nach der Festschreibung von ALTER TABLESPACE verwenden die in der Anweisung ALTER TABLESPACE angegebenen neuen Tabellenbereichsattribute. Die Unterstützung für *vorläufige Inaktivierung* ist auf dynamisches SQL beschränkt. Statische SQL-Abhängigkeiten müssen für die neuen Werte manuell erkannt und erneut kompiliert werden.

Tabellenbereiche, die dieselbe Speichergruppe verwenden, können unterschiedliche Werte für PAGESIZE und EXTENTSIZE aufweisen. Diese Attribute beziehen sich auf die Tabellenbereichsdefinition und nicht auf die Speichergruppe.

Vorgehensweise

Geben Sie die folgende Anweisung ein, um einen Tabellenbereich einer Speichergruppe zuzuordnen:

```
CREATE TABLESPACE tbber USING STOGROUP
speichergruppe
```

Dabei ist *tbber* der neue Tabellenbereich und *speichergruppe* die zugeordnete Speichergruppe.

Szenario: Versetzen eines Tabellenbereichs in eine neue Speichergruppe

Das folgende Szenario zeigt, wie ein Tabellenbereich aus einer Speichergruppe in eine andere Speichergruppe versetzt werden kann.

In diesem Szenario wird davon ausgegangen, dass sich die Tabellenbereichsdaten in Containern befinden, die in den Speicherpfaden einer Speichergruppe enthalten sind. Die Tabellenbereichsdaten werden mit der Anweisung ALTER TABLESPACE in die neue Speichergruppe versetzt.

Beim Versetzen des Tabellenbereichs in die neue Speichergruppe werden die Container in der alten Speichergruppe in den Status 'Löschen anstehend' versetzt. Nach dem Festschreiben der Anweisung ALTER TABLESPACE werden die Container den Speicherpfaden der neuen Speichergruppe zugeordnet, die in den alten Speichergruppen enthaltenen Container werden in den Status 'Löschen anstehend' versetzt, und eine implizite REBALANCE-Operation wird eingeleitet. Mit dieser Operation werden dem neuen Speicherpfad Container zugeordnet und die Daten aus den vorhandenen Containern auf die neuen Container neu verteilt. Die Anzahl und die Größe der zu erstellenden Container hängt von der Anzahl der Speicherpfade in der Zielspeichergruppe und der Menge des freien Speicherplatzes in den neuen Speicherpfaden ab. Nach dem Versetzen sämtlicher Daten werden die alten Container gelöscht.

Das folgende Diagramm stellt beispielhaft dar, wie ein Tabellenbereich aus einer Speichergruppe in eine andere Speichergruppe versetzt wird. Dabei werden die folgenden Aktionen ausgeführt:

1. Den Speicherpfaden der Zielspeichergruppe werden neue Container zugeordnet.

2. Alle ursprünglichen Container werden in den Status 'Löschen anstehend' versetzt; neue Zuordnungsanforderungen werden von den neuen Containern verarbeitet.
3. Eine regressive Neuverteilung wird ausgeführt, wobei Daten aus den Containern in den Pfaden, die gelöscht werden, entfernt und in andere Container versetzt werden.
4. Die Container werden physisch gelöscht.

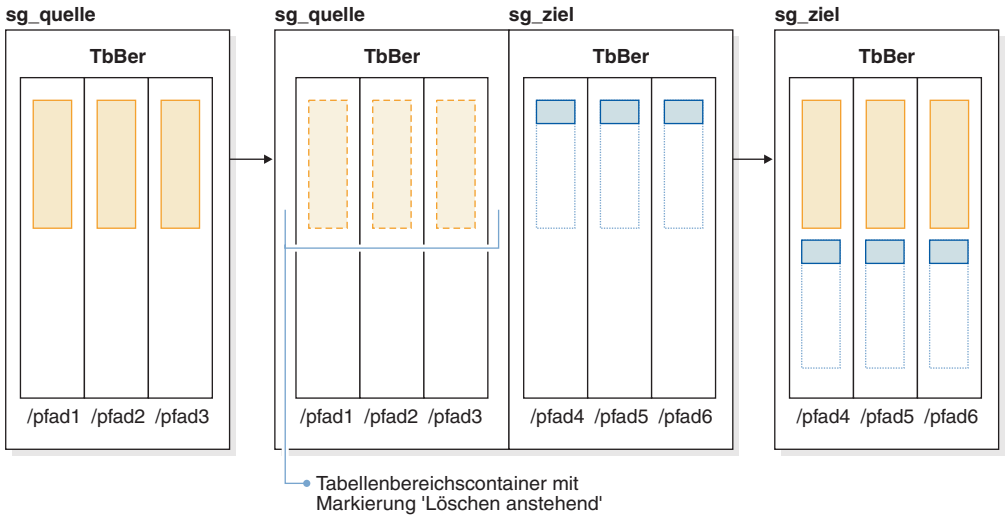


Abbildung 33. Versetzen eines Tabellenbereichs in eine neue Speichergruppe

Gehen Sie wie folgt vor, um einen Tabellenbereich in eine andere Speichergruppe zu versetzen:

1. Erstellen Sie zwei Speichergruppen, 'sg_quelle' und 'sg_ziel':
   ```
   CREATE STOGROUP sg_quelle ON '/pfad1', '/pfad2', '/pfad3'
   CREATE STOGROUP sg_ziel ON '/pfad4', '/pfad5', '/pfad6'
   ```
2. Erstellen Sie nach der Erstellung der Datenbank einen Tabellenbereich mit dynamischem Speicher, der anfänglich die Speichergruppe 'sg_quelle' verwendet:
   ```
   CREATE TABLESPACE tbber USING STOGROUP sg_quelle
   ```
3. Versetzen Sie den Tabellenbereich mit dynamischem Speicher in die Speichergruppe 'sg_ziel':
   ```
   ALTER TABLESPACE tbber USING sg_ziel
   ```

Kapitel 10. Schemata

Ein *Schema* ist eine Sammlung von benannten Objekten. Es stellt eine Möglichkeit zur logischen Gruppierung dieser Objekte bereit. Ein Schema ist außerdem ein Qualifikationsmerkmal für Namen. Es stellt eine Möglichkeit zur Verwendung desselben natürlichen Namens für mehrere Objekte sowie zur Vermeidung mehrdeutiger Verweise auf diese Objekte bereit.

Beispiel: Mithilfe der Schemanamen 'INTERNAL' und 'EXTERNAL' ist die Unterscheidung zweier unterschiedlicher SALES-Tabellen (INTERNAL.SALES, EXTERNAL.SALES) einfach.

Mit Schemata können auch mehrere Anwendungen Daten in einer einzigen Datenbank speichern, ohne dass es zu Namensbereichskollisionen kommt.

Ein Schema unterscheidet sich von einem *XML-Schema* und darf mit einem solchen nicht verwechselt werden. Ein XML-Schema ist ein Standard, der die Struktur beschreibt und den Inhalt von XML-Dokumenten prüft.

Ein Schema kann Tabellen, Sichten, Kurznamen, Trigger, Funktionen, Pakete und weitere Objekte enthalten. Ein Schema selbst ist ein Datenbankobjekt. Es wird explizit mit der Anweisung CREATE SCHEMA erstellt, wobei der aktuelle Benutzer oder eine angegebene Berechtigungs-ID als Schemaeigner erfasst wird. Es kann auch implizit beim Erstellen eines anderen Objekts erstellt werden, wenn der Benutzer über die Berechtigung IMPLICIT_SCHEMA verfügt.

Ein *Schemaname* wird als höherwertige Komponente eines zweiteiligen Objektnamens verwendet. Wenn das Objekt bei der Erstellung speziell mit einem Schemanamen qualifiziert wird, wird das Objekt dem Schema zugeordnet. Wenn bei der Erstellung des Objekts kein Schemaname angegeben wird, wird der Standardschemaname verwendet (angegeben im Sonderregister CURRENT SCHEMA).

Beispiel: Ein Benutzer mit der Berechtigung DBADM erstellt ein Schema mit dem Namen C für den Benutzer A:

```
CREATE SCHEMA C AUTHORIZATION A
```

Der Benutzer A kann dann die folgende Anweisung zum Erstellen einer Tabelle mit dem Namen X im Schema C absetzen (vorausgesetzt, dass der Benutzer A über die Datenbankberechtigung CREATETAB verfügt):

```
CREATE TABLE C.X (COL1 INT)
```

Einige Schemanamen sind reserviert. Integrierte Funktionen z. B. gehören zum Schema SYSIBM, und die vorinstallierten benutzerdefinierten Funktionen gehören zum Schema SYSFUN.

Wenn eine Datenbank erstellt wird, verfügen alle Benutzer über die Berechtigung IMPLICIT_SCHEMA, sofern sie nicht mit der Option `RESTRICTIVE` erstellt wird. Mit dieser Berechtigung erstellen Benutzer implizit ein Schema, wenn sie ein Objekt mit einem Schemanamen erstellen, der nicht bereits existiert. Bei der impliziten Erstellung von Schemata werden CREATEIN-Zugriffsrechte erteilt, mit denen alle Benutzer weitere Objekte in diesem Schema erstellen können. Die Funktionalität, Objekte wie z. B. Aliasnamen, einzigartige Datentypen, Funktionen und Trigger zu

erstellen, wurde auf implizit erstellte Schemata ausgeweitet. Die Standardzugriffsrechte für ein implizit erstelltes Schema stellen die Abwärtskompatibilität mit älteren Versionen bereit.

Wenn PUBLIC die Berechtigung IMPLICIT_SCHEMA entzogen wird, können Schemata mithilfe der Anweisung CREATE SCHEMA explizit bzw. durch Benutzer implizit erstellt werden (z. B. durch die Benutzer mit der Berechtigung DBADM), denen die Berechtigung IMPLICIT_SCHEMA erteilt wurde. Zwar wird durch das Entziehen der Berechtigung IMPLICIT_SCHEMA von PUBLIC die Kontrolle über die Verwendung von Schemanamen verschärft, aber es kann zu Berechtigungsfehlern kommen, wenn vorhandene Anwendungen Objekte erstellen.

Schemata verfügen ebenfalls über Zugriffsrechte, mit denen Schemaeigner steuern können, welche Benutzer das Zugriffsrecht zum Erstellen, Ändern, Kopieren und Löschen von Objekten in dem Schema haben. Dadurch kann die Bearbeitung einer Untergruppe von Objekten in der Datenbank gesteuert werden. Ein Schemaeigner erhält anfangs alle diese Zugriffsrechte für das Schema zusammen mit der Möglichkeit, diese Zugriffsrechte anderen Personen zu erteilen. Ein implizit erstelltes Schema gehört dem System, und alle Benutzer erhalten anfangs das Zugriffsrecht zum Erstellen von Objekten in einem solchen Schema. Ein Benutzer mit der Berechtigung ACCESSCTRL oder SECADM kann die Zugriffsrechte ändern, über die Benutzer für ein beliebiges Schema verfügen. Somit kann der Zugriff zum Erstellen, Ändern, Kopieren und Löschen von Objekten in einem beliebigen Schema (auch in einem implizit erstellten Schema) gesteuert werden.

Entwerfen von Schemata

Wenn Sie Ihre Daten in Tabellen organisieren, kann es vorteilhaft sein, die Tabellen und andere andere zugehörige Objekte in Gruppen zusammenzufassen. Dies geschieht durch Definieren eines Schemas mithilfe der Anweisung CREATE SCHEMA.

Informationen zu dem Schema werden in den Systemkatalogtabellen der Datenbank gespeichert, mit der Sie verbunden sind. Wenn weitere Objekte erstellt werden, können diese in den Schemata, die Sie erstellen, angelegt werden. Es ist jedoch zu beachten, dass sich ein Objekt jeweils nur in einem Schema befinden kann.

Schemata lassen sich mit Verzeichnissen vergleichen, wobei das aktuelle Schema dem aktuellen Verzeichnis entspricht. Nach dieser Analogie entspricht die Anweisung SET SCHEMA dem Befehl 'CD' (**Change Directory** - Verzeichnis wechseln).

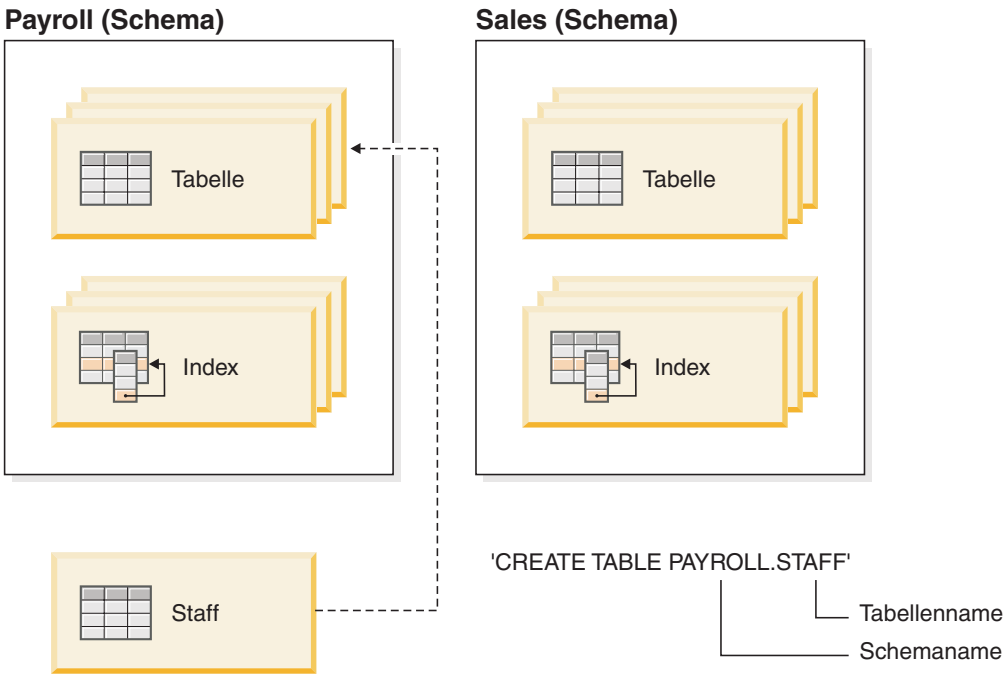
Wichtig: Es ist wichtig zu wissen, dass es keine Beziehung zwischen Berechtigungs-IDs und Schemata gibt, mit Ausnahme der Standardeinstellung CURRENT SCHEMA (im folgenden Abschnitt beschrieben).

Beim Entwerfen Ihrer Datenbanken und Tabellen sollten Sie sich außerdem Gedanken zu den Schemata in Ihrem System machen. Dazu gehören zum Beispiel die Auswahl von Namen für die Schemata sowie die Objekte, die den einzelnen Schemata zugeordnet werden sollen.

Den meisten Objekten in einer Datenbank wird ein eindeutiger Name zugeordnet, der aus zwei Teilen besteht. Der erste (linke) Teil wird als Qualifikationsmerkmal (Qualifier) oder Schema bezeichnet. Der zweite (rechte) Teil ist der einfache (bzw. unqualifizierte) Name. Syntaktisch werden diese beiden Teile zu einer, durch einen Punkt getrennten Zeichenfolge verkettet. Wenn ein Objekt, das durch einen Schemanamen qualifiziert werden kann (z. B. Tabelle, Index, Sicht, benutzerdefinierter

Datentyp, benutzerdefinierte Funktion, Kurzname, Paket oder Trigger) zu Anfang erstellt wird, wird ihm ein bestimmtes Schema auf der Grundlage des Qualifikationsmerkmals in seinem Namen zugeordnet.

Das folgende Diagramm veranschaulicht zum Beispiel, wie eine Tabelle während des Prozesses der Tabellenerstellung einem bestimmten Schema zugeordnet wird:



Sie sollten außerdem wissen, wie der Schemazugriff erteilt wird, um Ihren Benutzern die entsprechende Berechtigung und zutreffende Anweisungen geben zu können:

Schemanamen

Wenn ein neues Schema erstellt wird, darf der Name weder mit einem bereits im Katalog beschriebenen Schemanamen identisch sein noch mit der Buchstabenfolge "SYS" beginnen. Weitere Einschränkungen und Empfehlungen finden Sie in „Einschränkungen und Empfehlungen zu Schemanamen” auf Seite 302.

Zugriff auf Schemata

Ein Zugriff ohne Angabe eines Schemas als Qualifikationsmerkmal auf Objekte in einem Schema ist nicht zulässig, da durch das Schema die Eindeutigkeit in der Datenbank sichergestellt wird. Angesichts der Möglichkeit, dass zwei Benutzer zwei Tabellen (oder andere Objekte) mit demselben Namen erstellen könnten, leuchtet dies ein. Ohne Schema, das für Eindeutigkeit sorgt, gäbe es Mehrdeutigkeiten, wenn ein dritter Benutzer versuchte, die Tabelle abzufragen. Ohne weitere Qualifikationsmerkmale ist es unmöglich, die zu verwendende Tabelle zu ermitteln.

Der Benutzer, der durch die Anweisung CREATE SCHEMA erstellte Objekte definiert hat, ist der Eigner des Schemas. Dieser Eigner kann Zugriffsrechte auf Schemata anderen Benutzern erteilen (GRANT) und entziehen (REVOKE).

Wenn ein Benutzer die Berechtigung DBADM hat, kann er ein Schema mit einem beliebigen gültigen Namen erstellen. Bei der Erstellung einer Datenbank wird die Berechtigung IMPLICIT_SCHEMA der Gruppe PUBLIC (d. h. allen Benutzern) erteilt.

Wenn Benutzer nicht über die Berechtigung IMPLICIT_SCHEMA oder DBADM verfügen, können sie nur ein Schema erstellen, das den gleichen Namen wie ihre eigene Berechtigungs-ID hat.

Standardschema

Wenn kein Schema oder Qualifikationsmerkmal als Teil des Namens des zu erstellenden Objekts angegeben wird, wird dieses Objekt dem Standardschema zugeordnet, das im Sonderregister CURRENT SCHEMA angegeben ist. Der Standardwert dieses Sonderregisters ist der Wert der Berechtigungs-ID der Sitzung.

Ein Standardschema wird von nicht qualifizierten Objektverweisen in dynamischen Anweisungen benötigt. Sie können ein Standardschema für eine bestimmte DB2-Verbindung festlegen, indem Sie das Sonderregister CURRENT SCHEMA auf das Schema setzen, das als Standardschema verwendet werden soll. Zum Festlegen dieses Sonderregisters ist keine bestimmte Berechtigung erforderlich, sodass jeder beliebige Benutzer das Sonderregister CURRENT SCHEMA festlegen kann.

Die Syntax der Anweisung SET SCHEMA sieht wie folgt aus:

```
SET SCHEMA = <schemaname>
```

Sie können diese Anweisung interaktiv oder aus einer Anwendung heraus absetzen. Der Anfangswert des Sonderregisters CURRENT SCHEMA entspricht der Berechtigungs-ID des Benutzers der aktuellen Sitzung. Weitere Informationen zu diesem Thema finden Sie in den Informationen zur Anweisung SET SCHEMA.

Anmerkung:

- Es gibt darüber hinaus noch weitere Methoden, um das Standardschema beim Verbindungsaufbau festzulegen. Dazu können zum Beispiel die Datei `cli.ini` für CLI/ODBC-Anwendungen oder die Verbindungseigenschaften für die JDBC-Programmierschnittstelle (Java Database Connectivity) verwendet werden.
- Der Standardschemasatz wird nicht in den Systemkatalogen erstellt, sondern ist nur als Wert vorhanden, den der Datenbankmanager jederzeit (aus dem Sonderregister CURRENT SCHEMA) abrufen kann, wenn kein Schema oder Qualifikationsmerkmal als Teil des Namens eines zu erstellenden Objekts angegeben wird.

Implizite Erstellung

Sie können Schemata implizit erstellen, wenn Sie über die Berechtigung IMPLICIT_SCHEMA verfügen. Mit dieser Berechtigung können Sie ein Schema implizit erstellen, wenn Sie ein Objekt mit einem Schemanamen erstellen, der noch nicht vorhanden ist. Häufig werden Schemata implizit erstellt, wenn zum ersten Mal ein Datenobjekt in einem Schema erstellt wird, sofern der Benutzer, der das Objekt erstellt, die Berechtigung IMPLICIT_SCHEMA besitzt.

Explizite Erstellung

Schemata können darüber hinaus auch explizit erstellt und gelöscht werden. Dazu muss die Anweisung CREATE SCHEMA bzw. DROP SCHEMA

über die Befehlszeile oder aus einem Anwendungsprogramm heraus ausgeführt werden. Weitere Informationen zu diesem Thema finden Sie in den Informationen zu den Anweisungen CREATE SCHEMA und DROP SCHEMA.

Nach Schema definierte Aliasnamen für Tabellen und Sichten

Wenn ein anderer Benutzer in der Lage sein soll, auf eine Tabelle oder Sicht zuzugreifen, ohne den entsprechenden Schemanamen als Teil der Qualifikation des Tabellen- oder Sichtnamens einzugeben, muss ein Aliasname für diesen Benutzer erstellt werden. Die Definition des Aliasnamens muss den vollständig qualifizierten Tabellen- oder Sichtnamen einschließlich des Schemas des Benutzers definieren. Anschließend führt der Benutzer seine Abfrage unter Verwendung des Aliasnamens aus. Der Aliasname muss durch das Schema des Benutzers als Teil der Aliasnamensdefinition vollständig qualifiziert werden.

Gruppieren von Objekten nach Schema

Datenbankobjektnamen können aus einer einzigen Kennung bestehen; sie können aber auch *über ein Schema qualifizierte Objekte* mit zwei Kennungen sein. Das Schema, d. h. die höherwertige Komponente, eines derart qualifizierten Objekts stellt eine Methode bereit, Objekte in der Datenbank zu klassifizieren oder zu gruppieren. Wenn ein Objekt wie eine Tabelle, eine Sicht, ein Aliasname, ein benutzerdefinierter Datentyp, eine Funktion, ein Index, ein Paket oder ein Trigger erstellt wird, wird es einem Schema zugeordnet. Diese Zuordnung erfolgt entweder explizit oder implizit.

Eine explizite Verwendung des Schemas liegt vor, wenn Sie die höherwertige Komponente eines zweiteiligen Objektnamens beim Verweisen auf das Objekt in einer Anweisung verwenden. Der Benutzer A führt zum Beispiel eine Anweisung CREATE TABLE in Schema C wie folgt aus:

```
CREATE TABLE C.X (COL1 INT)
```

Eine implizite Verwendung des Schemas liegt vor, wenn Sie die höherwertige Komponente eines zweiteiligen Objektnamens nicht verwenden. Wenn dies der Fall ist, wird anhand des Sonderregisters CURRENT SCHEMA der Schemaname ermittelt, der als höherwertige Komponente des Objektnamens ergänzt wird. Der Anfangswert von CURRENT SCHEMA ist die Berechtigungs-ID des Benutzers der aktuellen Sitzung. Wenn Sie diesen Wert während der aktuellen Sitzung ändern möchten, können Sie für das Sonderregister mit der Anweisung SET SCHEMA einen anderen Schemanamen definieren.

Bei der Erstellung der Datenbank werden einige Objekte innerhalb bestimmter Schemata erstellt und in den Systemkatalogtabellen gespeichert.

Sie müssen nicht explizit angeben, in welchem Schema ein Objekt erstellt werden soll; wenn es nicht angegeben ist, wird die Berechtigungs-ID der Anweisung verwendet. Beispiel: Für die folgende CREATE TABLE-Anweisung nimmt der Schemaname standardmäßig den Wert der Berechtigungs-IDs an, die momentan angemeldet ist (d. h., der Wert des Sonderregisters CURRENT SCHEMA):

```
CREATE TABLE X (COL1 INT)
```

Bei dynamischen SQL- und XQuery-Anweisungen wird in der Regel der Wert des Sonderregisters CURRENT SCHEMA verwendet, um implizit sämtliche Objektnamenverweise ohne Qualifikationsmerkmal zu qualifizieren.

Bevor Sie eigene Objekte erstellen, müssen Sie entscheiden, ob Sie sie im eigenen Schema erstellen wollen oder ob Sie ein anderes Schema verwenden wollen, das die Objekte logisch gruppiert. Wenn Objekte erstellt werden, die gemeinsam benutzt werden sollen, kann die Verwendung eines anderen Schemanamens sehr vorteilhaft sein.

Einschränkungen und Empfehlungen zu Schemanamen

Bei der Benennung von Schemata müssen Sie einige Einschränkungen und Empfehlungen berücksichtigen.

- Benutzerdefinierte Typen (UDTs) dürfen keine Schemanamen verwenden, deren Längen die in „SQL- und XML-Begrenzungen" (SQL and XML limits) im Handbuch *SQL Reference* aufgeführte Schemalänge überschreiten.
- Die folgenden Schemanamen sind reservierte Wörter und dürfen nicht verwendet werden: SYSCAT, SYSFUN, SYSIBM, SYSSTAT, SYSPROC.
- Um mögliche Probleme bei Upgrades von Datenbanken in der Zukunft zu vermeiden, sollten Sie keine Schemanamen verwenden, die mit der Zeichenfolge SYS beginnen. Der Datenbankmanager lässt die Erstellung von Modulen, Prozeduren, Triggern, benutzerdefinierten Typen oder benutzerdefinierten Funktionen, die einen mit SYS beginnenden Schemanamen verwenden, nicht zu.
- Es wird empfohlen, das Wort SESSION nicht als Schemanamen zu verwenden. Deklarierte temporäre Tabellen müssen durch SESSION qualifiziert werden. Daher kann es vorkommen, dass eine Anwendung eine temporäre Tabelle mit einem Namen deklariert, der mit dem einer persistenten Tabelle identisch ist. In diesem Fall kann die Anwendungslogik zu komplex werden. Vermeiden Sie die Verwendung des Schemas SESSION, außer wenn Sie mit deklarierten temporären Tabellen arbeiten.

Erstellen von Schemata

Mithilfe von Schemata können Sie Objekte bei ihrer Erstellung gruppieren. Ein Objekt kann nur zu einem Schema gehören. Zur Erstellung von Schemata verwenden Sie die Anweisung CREATE SCHEMA.

Informationen zu den Schemata werden in den Systemkatalogtabellen der Datenbank gespeichert, zu der die Verbindung hergestellt wurde.

Vorbereitende Schritte

Wenn Sie ein Schema erstellen und einen anderen Benutzer optional zum Eigner dieses Schemas machen möchten, benötigen Sie dazu die Berechtigung DBADM. Wenn Sie die Berechtigung DBADM nicht haben, können Sie dennoch ein Schema mit Ihrer eigenen Berechtigungs-ID erstellen. Der Benutzer, der durch die Anweisung CREATE SCHEMA erstellte Objekte definiert hat, ist der Eigner des Schemas. Dieser Eigner kann Schemazugriffsrechte anderen Benutzern erteilen (GRANT) und entziehen (REVOKE).

Vorgehensweise

Geben Sie die folgende Anweisung in die Befehlszeile ein, um ein Schema zu erstellen:

```
CREATE SCHEMA schemaname [ AUTHORIZATION name_des_schemaeigners ]
```

Dabei ist *schemaname* der Name des Schemas. Dieser Name muss unter den bereits im Katalog erfassten Schemata eindeutig sein und darf nicht mit der Zeichenfolge

SYS beginnen. Wenn die optionale Klausel AUTHORIZATION angegeben wird, wird die in *name_des_schemaeigners* angegebene Berechtigungs-ID Eigner des Schemas. Fehlt diese Klausel, wird die Berechtigungs-ID, unter der diese Anweisung abgesetzt wurde, zum Eigner des Schemas.
Weitere Informationen finden Sie in der Beschreibung der Anweisung CREATE SCHEMA. Siehe auch „Einschränkungen und Empfehlungen zu Schemanamen" auf Seite 302.

Kopieren von Schemata

Das Dienstprogramm **db2move** und die Prozedur ADMIN_COPY_SCHEMA ermöglichen Ihnen die rasche Erstellung von Kopien eines Datenbankschemas. Nach der Einrichtung eines Modellschemas können Sie dieses als Vorlage für die Erstellung neuer Versionen verwenden.

Vorgehensweise

- Verwenden Sie die Prozedur ADMIN_COPY_SCHEMA, um ein einzelnes Schema innerhalb derselben Datenbank zu kopieren.
- Verwenden Sie das Dienstprogramm **db2move** mit der **-co COPY**, um ein einzelnes Schema oder mehrere Schemas aus einer Quellendatenbank in eine Zieldatenbank zu kopieren. Die meisten Datenbankobjekte aus dem Quellenschema werden unter das neue Schema in der Zieldatenbank kopiert.

Tipps zur Fehlerbehebung

Sowohl die Prozedur ADMIN_COPY_SCHEMA als auch das Dienstprogramm **db2move** rufen den Befehl **LOAD** auf. Während der Verarbeitung des Ladevorgangs werden die Tabellenbereiche, in denen sich die Zieldatenbankobjekte befinden, in den Status 'Backup anstehend' versetzt.

Prozedur ADMIN_COPY_SCHEMA

Durch die Verwendung dieser Prozedur mit der Option COPYNO werden die Tabellenbereiche, in denen sich das Zielobjekt befindet, in den Status 'Backup anstehend' versetzt, wie in der obigen Anmerkung beschrieben. Um den Tabellenbereich aus dem Status 'Festlegen der Integrität anstehend' herauszunehmen, setzt diese Prozedur eine Anweisung SET INTEGRITY ab. In Situationen, in denen ein Zieltabellenobjekt definierte referenzielle Integritätsbedingungen besitzt, wird die Zieltabelle ebenfalls in den Status 'Festlegen der Integrität anstehend' versetzt. Da sich die Tabellenbereiche bereits im Status 'Backup anstehend' befinden, schlägt der Versuch der Prozedur ADMIN_COPY_SCHEMA zum Absetzen der Anweisung SET INTEGRITY fehl.

Zur Behebung dieses Problems müssen Sie einen Befehl **BACKUP DATABASE** absetzen, um den Status 'Backup anstehend' für die betroffenen Tabellenbereiche aufzuheben. Als Nächstes müssen Sie die Spalte `Statement_text` der Fehlertabelle prüfen, die von dieser Prozedur generiert wurde, um eine Liste der Tabellen im Status 'Festlegen der Integrität anstehend' zu finden. Führen Sie anschließend die Anweisung SET INTEGRITY für jede in der Liste aufgeführte Tabelle aus, um jede einzelne Tabelle aus dem Status 'Festlegen der Integrität anstehend' herauszunehmen.

Dienstprogramm 'db2move'

Dieses Dienstprogramm versucht, alle zulässigen Schemaobjekte mit Ausnahme der folgenden Typen zu kopieren:

- Tabellenhierarchie

- Zwischenspeichertabellen (vom Dienstprogramm LOAD in Umgebungen mit mehreren Datenbankpartitionen nicht unterstützt)
- JAR-Dateien (Java Routine Archives)
- Kurznamen
- Pakete
- Sichthierarchien
- Objektzugriffsrechte (Alle neuen Objekte werden mit Standardberechtigungen erstellt.)
- Statistiken (Neue Objekte enthalten keine statistischen Informationen.)
- Indexerweiterungen (in Verbindung mit benutzerdefinierten strukturierten Typen)
- Benutzerdefinierte strukturierte Typen und ihre Umsetzungsfunktionen

Fehler aufgrund nicht unterstützter Typen
Wenn ein Objekt eines der nicht unterstützten Typen im Quellenschema angetroffen wird, wird ein Eintrag in einer Fehlerdatei protokolliert. Die Fehlerdatei gibt an, dass ein nicht unterstützter Objekttyp erkannt wurde. Die Operation COPY wird trotzdem erfolgreich ausgeführt. Der protokollierte Eintrag soll Sie über Objekte informieren, die durch diese Operation nicht kopiert wurden.

Nicht mit Schemata gekoppelte Objekte
Objekte, die nicht mit einem Schema gekoppelt sind, wie zum Beispiel Tabellenbereiche und Ereignismonitore, werden bei einer Schemakopieroperation nicht berücksichtigt. Sie müssen sie in der Zieldatenbank vor dem Aufrufen der Schemakopieroperation erstellen.

Replizierte Tabellen
Wenn eine replizierte Tabelle kopiert wird, ist die neue Kopie der Tabelle nicht für die Replikation eingerichtet. Die Tabelle wird als reguläre Tabelle erneut erstellt.

Unterschiedliche Instanzen
Die Quellendatenbank muss katalogisiert werden, wenn sie sich nicht in derselben Instanz wie die Zieldatenbank befindet.

Option SCHEMA_MAP
Wenn die Option SCHEMA_MAP zur Angabe eines anderen Schemanamens in der Zieldatenbank verwendet wird, führt die Schemakopieroperation nur eine minimale Analyse der Objektdefinitionsanweisungen durch, um den ursprünglichen Schemanamen durch den neuen Schemanamen zu ersetzen. Zum Beispiel werden alle Vorkommen des ursprünglichen Schemanamens, die im Inhalt einer SQL-Prozedur auftreten, nicht durch den neuen Schemanamen ersetzt. Aus diesem Grund könnte die Erstellung dieser Objekte durch die Schemakopieroperation fehlschlagen. Andere Beispiele können möglicherweise die Zwischenspeichertabelle, die Ergebnistabelle und die MQT (Materialized Query Table) sein. Sie können die Anweisungen der Datendefinitionssprache (DDL) in der Fehlerdatei verwenden, um diese nicht erstellten Objekte nach Abschluss der Kopieroperation erneut zu erstellen.

Gegenseitige Abhängigkeiten zwischen Objekten
Die Schemakopieroperation versucht, Objekte in einer Reihenfolge erneut zu erstellen, die die gegenseitigen Abhängigkeiten zwischen diesen Objekten berücksichtigt. Wenn zum Beispiel eine Tabelle T1 eine Spalte enthält, die eine benutzerdefinierte Funktion U1 referenziert, erstellt die Operation die Funktion U1 erneut, bevor sie die Tabelle T1 erneut erstellt. Informatio-

nen zu Abhängigkeiten von Prozeduren sind jedoch in den Katalogen nicht einfach verfügbar. Bei der erneuten Erstellung von Prozeduren versucht die Schemakopieroperation daher zunächst, alle Prozeduren erneut zu stellen. Anschließend wiederholt sie den Versuch für die Prozeduren, deren erste Neuerstellung fehlgeschlagen ist. (Dies geschieht in der Annahme, dass sich Prozeduren, die von einer Prozedur abhängig sind, die beim vorherigen Versuch erfolgreich erstellt wurde, bei einem nachfolgenden Versuch erfolgreich neu erstellen lassen.) Die Operation setzt die Versuche, diese Prozeduren, deren Neuerstellung fehlgeschlagen ist, erneut zu erstellen, so lange fort, wie sie in der Lage ist, mindestens eine weitere Prozedur bei einem nachfolgenden Versuch erneut zu erstellen. Bei jedem Versuch, eine Prozedur erneut zu erstellen, wird ein Fehler (mit zugehörigen DDL-Anweisungen) in der Fehlerdatei protokolliert. Möglicherweise sehen Sie in der Fehlerdatei viele Einträge für dieselben Prozeduren. Diese Prozeduren können dennoch bei einem nachfolgenden Versuch erfolgreich erneut erstellt worden sein. Sie sollten nach Abschluss der Schemakopieroperation die Tabelle SYSCAT.PROCEDURES abfragen, um festzustellen, ob diese in der Fehlerdatei aufgeführten Prozeduren erfolgreich erneut erstellt wurden.

Weitere Informationen finden Sie in den Beschreibungen zur Prozedur ADMIN_COPY_SCHEMA und zum Dienstprogramm **db2move**.

Beispiel für das Kopieren eines Schemas mit der Prozedur ADMIN_COPY_SCHEMA

Zum Kopieren eines einzelnen Schemas innerhalb derselben Datenbank können Sie die im folgenden Beispiel gezeigte Prozedur ADMIN_COPY_SCHEMA verwenden.

```
DB2 "SELECT SUBSTR(OBJECT_SCHEMA,1, 8)
  AS OBJECT_SCHEMA, SUBSTR(OBJECT_NAME,1, 15)
  AS OBJECT_NAME, SQLCODE, SQLSTATE, ERROR_TIMESTAMP, SUBSTR(DIAGTEXT,1, 80)
  AS DIAGTEXT, SUBSTR(STATEMENT,1, 80)
  AS STATEMENT FROM COPYERRSCH.COPYERRTAB"

CALL SYSPROC.ADMIN_COPY_SCHEMA('SOURCE_SCHEMA', 'TARGET_SCHEMA',
  'COPY', NULL, 'SOURCETS1 , SOURCETS2', 'TARGETTS1, TARGETTS2,
  SYS_ANY', 'ERRORSCHEMA', 'ERRORNAME')
```

Die Ausgabe aus dieser Anweisung SELECT entspricht dem folgenden Beispiel:

```
OBJECT_SCHEMA OBJECT_NAME     SQLCODE     SQLSTATE ERROR_TIMESTAMP
------------- --------------- ----------- -------- --------------------------
SALES         EXPLAIN_STREAM         -290 55039    2006-03-18-03.22.34.810346

DIAGTEXT
--------------------------------------------------------------------------------
[IBM][CLI Driver][DB2/LINUXX8664] SQL0290N  Der Zugriff auf einen Tabellenbereich
ist nicht zulässig.


STATEMENT
--------------------------------------------------------------------------------
set integrity for "SALES  "."ADVISE_INDEX" , "SALES"."ADVISE_MQT" , "SALES"."


1 Satz/Sätze ausgewählt.
```

Beispiele für das Kopieren eines Schemas mit dem Dienstprogramm 'db2move'

Zum Kopieren eines einzigen Schemas oder mehrerer Schemata aus einer Quellendatenbank in eine Zieldatenbank können Sie das Dienstprogramm **db2move** mit der

Aktion **COPY -co** verwenden. Nach der Einrichtung eines Modellschemas können Sie dieses als Vorlage für die Erstellung neuer Versionen verwenden.

Beispiel 1: Verwenden der Optionen für 'COPY -co'

Im folgenden Beispiel für die **db2move**-Optionen für **COPY -co** wird das Schema BAR aus der Datenbank 'sample' in die Zieldatenbank ('target') kopiert und in FOO umbenannt:

```
db2move sample COPY -sn BAR -co target_db  target schema_map
  "((BAR,FOO))" -u benutzer-id -p kennwort
```

Die neuen (Ziel-)Schemaobjekte werden mit den gleichen Objektnamen wie die Objekte im Quellenschema, jedoch mit dem Qualifikationsmerkmal 'target' erstellt. Es ist möglich, Kopien von Tabellen mit oder ohne Daten aus der Quellentabelle zu erstellen. Die Quellendatenbank und die Zieldatenbank können sich auf verschiedenen Systemen befinden.

Beispiel 2: Angeben der Namenszuordnungen für Tabellenbereiche bei der COPY-Operation

Das folgende Beispiel zeigt, wie bestimmte Namenszuordnungen für Tabellenbereiche angegeben werden, die bei einer **COPY**-Operation mit dem Dienstprogramm **db2move** anstelle der Tabellenbereiche aus dem Quellensystem verwendet werden sollen. Sie können das Schlüsselwort SYS_ANY angeben, um festzulegen, dass der Zieltabellenbereich mithilfe des Standardauswahlalgorithmus für Tabellenbereiche ausgewählt werden soll. In diesem Fall wählt das Dienstprogramm **db2move** alle verfügbaren Tabellenbereiche zur Verwendung als Zielobjekte aus:

```
db2move sample COPY -sn BAR -co target_db  target schema_map
  "((BAR,FOO))" tablespace_map "(SYS_ANY)" -u benutzer-id -p kennwort
```

Das Schlüsselwort SYS_ANY kann für alle Tabellenbereiche verwendet werden. Alternativ können Sie auch bestimmte Zuordnungen für einige der Tabellenbereiche und den Standardauswahlalgorithmus für die restlichen Tabellenbereiche angeben:

```
db2move sample COPY -sn BAR -co target_db  target schema_map "
  ((BAR,FOO))" tablespace_map "((TS1, TS2),(TS3, TS4), SYS_ANY)"
  -u benutzer-id -p kennwort
```

In diesem Beispiel wird angegeben, dass Tabellenbereich TS1 dem Tabellenbereich TS2 und Tabellenbereich TS3 dem Tabellenbereich TS4 zugeordnet werden, die restlichen Tabellenbereiche jedoch den Standardauswahlalgorithmus für Tabellenbereiche verwenden.

Beispiel 3: Ändern der Objekteigner nach der COPY-Operation

Nach einer erfolgreichen COPY-Operation können Sie den Eigner der neuen Objekte ändern, die im Zielschema erstellt wurden. Der Standardeigner der Zielobjekte ist der Benutzer, der die Verbindung hergestellt hat. Wenn diese Option angegeben wird, wird das Eigentumsrecht an einen neuen Eigner wie folgt übertragen:

```
db2move sample COPY -sn BAR -co target_db  target schema_map
  "((BAR,FOO))" tablespace_map "(SYS_ANY)" owner jrichards
  -u benutzer-id -p kennwort
```

Der neue Eigner der Zielobjekte ist 'jrichards'.

Das Dienstprogramm **db2move** muss auf dem Zielsystem aufgerufen werden, wenn sich das Quellenschema und das Zielschema auf verschiedenen Systemen befinden. Um Schemata aus einer Datenbank in eine andere kopieren zu können, muss für diese Aktion eine Liste mit durch Kommata

getrennten Schemanamen, die aus einer Quellendatenbank zu kopieren sind, sowie ein Zieldatenbankname eingegeben werden.

Zum Kopieren eines Schemas geben Sie **db2move** in eine Eingabeaufforderung des Betriebssystems wie folgt ein:

```
db2move dbname COPY -co COPY-optionen
 -u benutzer-id -p kennwort
```

Erneutes Starten einer fehlgeschlagenen Operation zum Kopieren eines Schemas

Fehler, die bei einer Kopieroperation (COPY) mit dem Dienstprogramm **db2move** auftreten, können abhängig vom Typ des kopierten Objekts oder von der Phase der Kopieroperation, in der die Kopieroperation fehlgeschlagen ist (d. h., entweder die Phase, in der Objekte erneut erstellt werden, oder die Phase, in der die Daten geladen werden), auf unterschiedliche Weise behandelt werden.

Informationen zu diesem Vorgang

Das Dienstprogramm **db2move** meldet Nachrichten und Fehler an den Benutzer in Nachrichten- und Fehlerdateien zurück. Schemakopieroperationen verwenden die Nachrichtendatei COPYSCHEMA_*zeitmarke*.MSG und die Fehlerdatei COPYSCHEMA_*zeitmarke*.err. Diese Dateien werden im aktuellen Arbeitsverzeichnis erstellt. Die aktuelle Uhrzeit wird an den Dateinamen angefügt, um die Eindeutigkeit der Dateien sicherzustellen. Dem Benutzer obliegt es, diese Nachrichten- und Fehlerdateien wieder zu löschen, wenn sie nicht mehr benötigt werden.

Anmerkung: Es ist möglich, mehrere Instanzen von **db2move** gleichzeitig auszuführen. Die Option COPY gibt keine SQLCODE-Werte zurück. Dies ist mit dem Verhalten des Dienstprogramms **db2move** konsistent.

Der Typ des Objekts, das gerade kopiert wird, lässt sich in eine von zwei Kategorien einordnen: physisches Objekt oder Geschäftsobjekt.

Die Bezeichnung physisches Objekt bezieht sich auf Objekte, die sich physisch in einem Container befinden, wie zum Beispiel Tabellen, Indizes und benutzerdefinierte strukturierte Typen. Die Bezeichnung Geschäftsobjekt bezieht sich auf katalogisierte Objekte, die sich nicht in Containern befinden, wie zum Beispiel Sichten, benutzerdefinierte strukturierte Typen (UDTs) und Aliasnamen.

Fehler, die während der erneuten Erstellung eines physischen Objekts auftreten, bewirken, dass das Dienstprogramm ein Rollback durchführt, wohingegen Fehler, die während der erneuten Erstellung eines logischen Objekts auftreten, kein Rollback auslösen.

Vorgehensweise

Gehen Sie wie folgt vor, um die Schemakopieroperation erneut zu starten:

Führen Sie nach dem Beheben der Probleme, die das Fehlschlagen der Ladeoperationen verursacht haben (siehe Fehlerdatei), den Befehl **db2move COPY** erneut aus. Verwenden Sie den Parameter **-tf** in Verbindung mit dem Dateinamen LOADTABLE.err, um die Tabellen anzugeben, die kopiert und mit Daten gefüllt werden sollen. Beispiel:

```
db2move quellen_db COPY -tf LOADTABLE.err -co TARGET_DB ziel_db
  -mode load_only
```

Sie können die Tabellennamen auch manuell mit dem Parameter **-tn** angeben. Beispiel:

```
db2move quellen_db COPY -tn "FOO"."TABELLE1","FOO 1"."TAB 444",
  -co TARGET_DB ziel_db -mode load_only
```

Anmerkung: Der Modus `load_only` erfordert die Eingabe mindestens einer Tabelle mit dem Parameter **-tn** oder **-tf**.

Beispiele

Beispiel 1: Fehler beim Kopieren von Schemata im Zusammenhang mit physischen Objekten

Fehler, die bei der Neuerstellung physischer Objekte in der Zieldatenbank auftreten, werden in der Fehlerdatei `COPYSCHEMA_timestamp.err` protokolliert. Für jedes fehlerhafte Objekt enthält die Fehlerdatei Informationen wie den Objektnamen, den Objekttyp, den DDL-Text, die Zeitmarke und einen als Zeichenfolge formatierten SQL-Kommunikationsbereich (SQLCA-Feldnamen gefolgt von den zugehörigen Datenwerten).

Beispielausgabe für die Fehlerdatei `COPYSCHEMA_`*`zeitmarke`*`.err`:

```
1. schema: FOO.T1
Type:      TABLE
Error Msg: SQL0104N Auf ... folgte das unerwartete Token 'FOO.T1'.
Timestamp: 2005-05-18-14.08.35.65
DDL:       create view FOO.v1

2. schema:  FOO.T3
Type:      TABLE
Error Msg: SQL0204N FOO.V1 ist ein nicht definierter Name.
Timestamp: 2005-05-18-14.08.35.68
DDL:       create table FOO.T3
```

Wenn Fehler bei der Erstellung physischer Objekte am Ende der Phase der Neuerstellung und vor der Ladephase protokolliert werden, schlägt die Ausführung des Dienstprogramms **db2move** unter Rückgabe eines Fehlers fehl. Alle bereits ausgeführten Schritte der Objekterstellung in der Zieldatenbank werden rückgängig gemacht (Rollback), während in der Quellendatenbank alle intern erstellten Tabellen bereinigt werden. Um alle potenziellen Fehler in der Fehlerdatei zu erfassen, erfolgt der Rollback am Ende der Neuerstellungsphase nach dem Versuch, die einzelnen Objekte erneut zu erstellen (nicht nach dem ersten Fehlschlag). Dies bietet Ihnen die Möglichkeit, alle Probleme zu beheben, bevor Sie die **db2move**-Operation erneut starten. Wenn keine Fehler festgestellt werden, wird die Fehlerdatei gelöscht.

Beispiel 2: Fehler beim Kopieren von Schemata im Zusammenhang mit Geschäftsobjekten

Fehler, die bei der Neuerstellung von Geschäftsobjekten in der Zieldatenbank auftreten, führen nicht dazu, dass die Ausführung des Dienstprogramms **db2move** fehlschlägt. Solche Fehler werden stattdessen in der Fehlerdatei `COPYSCHEMA_`*`zeitmarke`*`.err` protokolliert. Nach der Beendigung des Dienstprogramms **db2move** können Sie die Fehler untersuchen, entsprechende Maßnahmen ergreifen und jedes Objekt, dessen Erstellung fehlgeschlagen ist, manuell erneut erstellen. (Hilfreiche DDL-Anweisungen zu diesem Zweck werden in der Fehlerdatei bereitgestellt).

Ein Fehler, der auftritt, wenn das Dienstprogramm **db2move** versucht, Tabellendaten mithilfe des Dienstprogramms LOAD in Tabellen zu laden, führt nicht dazu, dass die Ausführung des Dienstprogramms **db2move** fehl-

schlägt. Vielmehr werden generische Fehlerinformationen in der Datei COPYSCHEMA_*zeitmarke*.err (z. B. Objektname, Objekttyp, DDL-Text, Zeitmarke, SQL-Kommunikationsbereich) protokolliert. Der vollständig qualifizierte Name der Tabelle wird in einer anderen Datei mit dem Namen LOADTABLE_*zeitmarke*.err protokolliert. In dieser Datei wird eine Tabelle pro Zeile aufgeführt, um die Formatanforderung des **db2move**-Parameters **-tf** zu erfüllen. Das Format sieht in etwa wie folgt aus:

```
"FOO"."TABLE1"
"FOO 1"."TAB 444"
```

Beispiel 3: Andere Typen von db2move-Fehlern

Fehler bei internen Operationen, wie zum Beispiel Speicherfehler oder Dateisystemfehler, können dazu führen, dass die Ausführung des Dienstprogramms **db2move** fehlschlägt.

Wenn während der Phase der Neuerstellung durch DDL-Anweisungen ein Fehler bei internen Operationen auftritt, erfolgt ein Rollback aller erfolgreich erstellten Objekte im Zielschema. Alle intern erstellten Tabellen, z. B. die DMT-Tabelle und die **db2look**-Tabelle, werden in der Quellendatenbank bereinigt.

Wenn während der Ladephase ein Fehler bei internen Operationen auftritt, verbleiben alle erfolgreich erstellten Objekte im Zielschema. Alle Tabellen, bei denen während einer Ladeoperation ein Fehler auftritt, und alle Tabellen, die noch nicht geladen wurden, werden in der Fehlerdatei LOADTABLE.err protokolliert. Sie können anschließend den Befehl **db2move COPY** mit der Datei LOADTABLE.err wie in Beispiel 2 gezeigt ausführen. Wenn das Dienstprogramm **db2move** abnormal beendet wird (z. B. durch einen Systemabsturz, einen Trap oder einen manuellen Abbruch des Dienstprogramms), gehen die Informationen dazu, welche Tabellen noch zu laden sind, verloren. In diesem Fall können Sie das Zielschema mithilfe der Prozedur ADMIN_DROP_SCHEMA löschen und den Befehl **db2move COPY** erneut absetzen.

Unabhängig davon, welcher Fehler bei dem Versuch auftritt, eine Operation zum Kopieren eines Schemas auszuführen, haben Sie immer die Möglichkeit, das Zielschema mit der Prozedur ADMIN_DROP_SCHEMA zu löschen. Sie können den Befehl **db2move COPY** anschließend erneut absetzen.

Löschen von Schemata

Verwenden Sie zum Löschen eines Schemas die Anweisung DROP.

Vorbereitende Schritte

Bevor Sie ein Schema löschen, müssen alle Objekte, die sich in diesem Schema befinden, gelöscht oder in ein anderes Schema versetzt werden.

Der Schemaname muss im Katalog vorhanden sein, wenn die Anweisung DROP ausgeführt wird. Ansonsten wird ein Fehler zurückgegeben.

Vorgehensweise

Geben Sie in die Befehlszeile Folgendes ein, um eine Schema zu löschen:

```
DROP SCHEMA name RESTRICT
```

Das Schlüsselwort RESTRICT erzwingt die Regel, dass im angegebenen Schema keine Objekte für das Schema definiert werden können, das aus der Datenbank gelöscht werden soll. Das Schlüsselwort RESTRICT ist nicht optional.

Beispiel

Im folgenden Beispiel wird das Schema "joeschma" gelöscht:

```
DROP SCHEMA joeschma RESTRICT
```

Teil 3. Datenbankobjekte

Die Gestaltung logischer Datenbanken besteht aus der Definition von Datenbankobjekten.

In einer DB2-Datenbank können die folgenden Datenbankobjekte erstellt werden:

- Tabellen
- Integritätsbedingungen
- Indizes
- Trigger
- Sequenzen
- Sichten
- Nutzungslisten

Diese Datenbankobjekte können mithilfe von grafischen Benutzerschnittstellen oder durch das explizite Ausführen von Anweisungen erstellt werden. Die zum Erstellen dieser Datenbankobjekte verwendeten Anweisungen werden DLL-Anweisungen (DLL = Data Definition Language, Datendefinitionssprache) genannt und verfügen in der Regel über die Schlüsselwörter CREATE oder ALTER als Präfix.

Für die Implementierung einer guten Datenbankgestaltung, die den derzeitigen Bedürfnissen der Datenspeicherung Ihres Geschäfts entspricht und gleichzeitig flexibel genug ist, um mit der Zeit Erweiterung und Wachstum Rechnung zu tragen, ist es wichtig, sich über die Funktionen und die Funktionalität im Klaren zu sein, die jedes einzelne Datenbankobjekt bereitstellt.

Kapitel 11. Allgemeine Konzepte für die meisten Datenbankobjekte

Aliasnamen

Ein *Aliasname* ist ein alternativer Name für ein Objekt wie zum Beispiel ein Modul, eine Tabelle oder einen anderen Aliasnamen. Er kann an allen Stellen zum Verweisen auf ein Objekt verwendet werden, an denen auf dieses Objekt direkt verwiesen werden kann.

Ein Aliasname darf nicht in allen Kontexten verwendet werden. Er darf beispielsweise nicht in der Prüfbedingung einer Prüfung auf Integritätsbedingung verwendet werden. Ein Aliasname kann nicht auf eine deklarierte temporäre Tabelle, jedoch auf eine erstellte temporäre Tabelle verweisen.

Ebenso wie andere Objekte kann ein Aliasname erstellt und gelöscht werden, und es können ihm Kommentare zugeordnet werden. Aliasnamen können auf andere Aliasnamen in einem Prozess verweisen, der als *Verketten* bezeichnet wird, solange kein rückbezüglicher Verweis entsteht. Für die Verwendung von Aliasnamen ist weder eine besondere Berechtigung noch ein besonderes Zugriffsrecht erforderlich. Der Zugriff auf das Objekt, auf das ein Aliasname verweist, erfordert jedoch die dem Objekt zugeordnete Berechtigung.

Wenn ein Aliasname als *öffentlicher* Aliasname ('public') definiert ist, kann er durch seinen unqualifizierten Namen angegeben werden, ohne dass er durch den aktuellen Standardschemanamen beeinflusst wird. Er kann außerdem mit dem Qualifikationsmerkmal SYSPUBLIC angegeben werden.

Synonym ist eine alternative Bezeichnung für Aliasname.

Weitere Informationen enthält der Abschnitt zu Aliasnamen in Kennungen (Bezeichnern) im Handbuch *SQL Reference Volume 1*.

Vorläufige Inaktivierung von Datenbankobjekten

Wenn die *vorläufige Inaktivierung* aktiv ist, kann ein Objekt gelöscht werden, auch wenn es von anderen aktiven Transaktionen verwendet wird. Transaktionen, die das gelöschte Objekt verwenden, können fortfahren. Allen neuen Transaktionen wird jedoch der Zugriff auf das gelöschte Objekt verweigert.

Alle im Cache gespeicherten Anweisungen und Pakete, die direkt oder indirekt auf das Objekt, das gelöscht oder geändert wird, verweisen, werden als ungültig markiert (und als *inaktiviert* bezeichnet). Die vorläufige Inaktivierung sorgt dafür, dass DDL-Anweisungen, die sich auf die Objekte, auf die verwiesen wird, auswirken, Wartezeiten vermeiden können, die ansonsten daraus resultieren, das zurzeit ausgeführte Anweisungen Sperren für die Objekte, auf die sie verweisen, aktiviert haben. Gleichzeitig kann jeder bereits aktive Zugriff unter Verwendung der im Cache enthaltenen Version des Objekts fortgesetzt werden, sodass Überschreitungen des Sperrzeitlimits nicht möglich sind.

Im Gegensatz dazu wird bei der *absoluten Inaktivierung* eine exklusive Sperre verwendet, wenn auf ein Objekt verwiesen wird. Dies stellt sicher, dass alle Prozesse

mit denselben Versionen von Objekten arbeiten und dass keine Zugriffe auf ein Objekt stattfinden, wenn es gelöscht wurde.

Die vorläufige Inaktivierung wird durch die Registrierdatenbankvariable **DB2_DDL_SOFT_INVAL** aktiviert. Diese Registrierdatenbankvariable ist standardmäßig auf den Wert ON gesetzt.

Die folgende Liste enthält die DDL-Anweisungen (DDL, Data Definition Language, Datendefinitionssprache), für die die vorläufige Inaktivierung unterstützt wird:

- ALTER TABLE...DETACH PARTITION
- CREATE OR REPLACE ALIAS
- CREATE OR REPLACE FUNCTION
- CREATE OR REPLACE TRIGGER
- CREATE OR REPLACE VIEW
- DROP ALIAS
- DROP FUNCTION
- DROP TRIGGER
- DROP VIEW

Anmerkung: In DB2 Version 9.7 Fixpack 1 und späteren Releases führt die Anweisung ALTER TABLE...DETACH PARTITION eine vorläufige Inaktivierung auf allen Isolationsstufen an im Cache zwischengespeicherten Anweisungen aus, die direkt oder indirekt auf eine partitionierte Tabelle verweisen. Eine nachfolgende asynchrone Task zur Aufhebung der Partitionszuordnung führt eine absolute Inaktivierung der zuvor vorläufig inaktivierten Anweisungen im Cache aus, bevor die Partition mit aufgehobener Zuordnung in eine eigenständige Tabelle konvertiert wird.

Die Registrierdatenbankvariable **DB2_DDL_SOFT_INVAL** hat keine Auswirkung auf die Inaktivierung, die durch die Anweisung ALTER TABLE...DETACH PARTITION ausgeführt wird.

Die Unterstützung für die vorläufige Inaktivierung gilt nur für dynamische SQL-Anweisungen sowie für Suchläufe, die unter den Isolationsstufen 'Cursorstabilität' (CS) und 'Nicht festgeschriebener Lesevorgang' (UR) ausgeführt werden. Bei der Anweisung ALTER TABLE...DETACH PARTITION betrifft die vorläufige Inaktivierung Suchoperationen unter allen Isolationsstufen.

Beispiel

Nehmen Sie zum Beispiel an, dass eine Sicht mit dem Namen VIEW1 vorhanden ist. Sie öffnen einen Cursor und führen die Anweisung `SELECT * from VIEW1` aus. Kurz darauf setzt der Datenbankadministrator den Befehl `DROP VIEW VIEW1` ab, um die Sicht VIEW1 aus der Datenbank zu löschen. Bei der absoluten Inaktivierung wird die Anweisung DROP VIEW gezwungen, auf eine exklusive Sperre für VIEW1 zu warten, bis die SELECT-Transaktion abgeschlossen ist. Bei der vorläufigen Inaktivierung wird der Anweisung DROP VIEW keine exklusive Sperre für die Sicht erteilt. Die Sicht wird gelöscht, jedoch kann die SELECT-Anweisung die Ausführung unter Verwendung der letzten Definition der Sicht fortsetzen. Wenn die SELECT-Anweisung abgeschlossen ist, führen alle nachfolgenden Versuche, die Sicht VIEW1 zu verwenden (selbst durch denselben Benutzer oder Prozess, der sie soeben noch verwendet hat), zu einem Fehler (SQL0204N).

Automatische Reaktivierung von Datenbankobjekten

Die *automatische Reaktivierung* ist ein Mechanismus, durch den Datenbankobjekte, die inaktiviert (ungültig gemacht) wurden, zum Beispiel durch eine DROP-Anweisung, eine automatische Reaktivierung (erneute Gültigmachung) durchlaufen.

Im Allgemeinen versucht der Datenbankmanager, ungültige Objekte wieder gültig zu machen, das heißt, zu reaktivieren, wenn diese Objekte das nächste Mal verwendet werden. Die automatische Reaktivierung wird durch den Konfigurationsparameter **auto_reval** aktiviert. Standardmäßig ist diese Registrierdatenbankvariable auf den Wert DEFERRED gesetzt. Bei Datenbanken, für die ein Upgrade von Version 9.5 oder einer früheren Version durchgeführt wurde, hat die Registrierdatenbankvariable **auto_reval** jedoch den Wert DISABLED.

Informationen zu den abhängigen Objekten, die betroffen sind, wenn ein Objekt gelöscht wird, sowie dazu, wann diese abhängigen Objekte reaktiviert werden, finden Sie in der Beschreibung der „Anweisung DROP" im Handbuch *SQL Reference Volume 1*.

Die folgende Liste enthält die DDL-Anweisungen (DDL, Data Definition Language, Datendefinitionssprache), für die die automatische Reaktivierung gegenwärtig unterstützt wird:

- ALTER MODULE DROP FUNCTION
- ALTER MODULE DROP PROCEDURE
- ALTER MODULE DROP TYPE
- ALTER MODULE DROP VARIABLE
- ALTER NICKNAME (zum Ändern des lokalen Namens oder Typs)
- ALTER TABLE ALTER COLUMN
- ALTER TABLE DROP COLUMN
- ALTER TABLE RENAME COLUMN
- CREATE OR REPLACE ALIAS
- CREATE OR REPLACE FUNCTION
- CREATE OR REPLACE NICKNAME
- CREATE OR REPLACE PROCEDURE
- CREATE OR REPLACE SEQUENCE
- CREATE OR REPLACE TRIGGER
- CREATE OR REPLACE VARIABLE
- CREATE OR REPLACE VIEW
- DROP FUNCTION
- DROP NICKNAME
- DROP PROCEDURE
- DROP SEQUENCE
- DROP TABLE
- DROP TRIGGER
- DROP TYPE
- DROP VARIABLE
- DROP VIEW
- RENAME TABLE

Mit der Prozedur ADMIN_REVALIDATE_DB_OBJECTS können Sie vorhandene Objekte reaktivieren, die als inaktiv gekennzeichnet wurden.

Erstellen und Verwalten von Datenbankobjekten

Bei der Erstellung einiger Typen von Datenbankobjekten sollten Sie mit der Unterstützung von CREATE mit Fehlern sowie mit der Option REPLACE vertraut sein.

Unterstützung von CREATE mit Fehlern für bestimmte Datenbankobjekte

Bestimmte Typen von Objekten können erstellt werden, selbst wenn bei ihrer Kompilierung Fehler auftreten. Zum Beispiel kann eine Sicht erstellt werden, wenn die Tabelle, auf die sie verweist, nicht vorhanden ist.

Solche Objekte bleiben ungültig, bis auf sie zugegriffen wird. Die Unterstützung von CREATE mit Fehlern erstreckt sich gegenwärtig auf Sichten und Inline-SQL-Funktionen (nicht kompilierte Funktionen). Dieses Funktionsmerkmal ist aktiviert, wenn der Datenbankkonfigurationsparameter **auto_reval** auf den Wert IMMEDIATE oder DEFERRED gesetzt ist.

Die Fehler, die bei der Objekterstellung toleriert werden, sind auf die folgenden Typen begrenzt:

- Beliebige Fehler bei der Namensauflösung. Beispiele: Eine Tabelle, auf die verwiesen wird, ist nicht vorhanden (SQLSTATE-Werte 42704, SQL0204N); eine Spalte, auf die verwiesen wird, ist nicht vorhanden (SQLSTATE-Werte 42703, SQL0206N); oder eine Funktion, auf die verwiesen wird, wurde nicht gefunden (SQLSTATE-Werte 42884, SQL0440N).
- Beliebige Fehler bei der Reaktivierung (erneuten Gültigmachung) verschachtelter Objekte. Ein Objekt, das erstellt wird, kann auf Objekte verweisen, die nicht gültig (bzw. inaktiviert) sind. Für diese ungültigen Objekte wird die Reaktivierung aufgerufen. Wenn die Reaktivierung eines ungültigen Objekts, auf das verwiesen wird, fehlschlägt, wird die Anweisung CREATE erfolgreich ausgeführt und das erstellte Objekt bleibt ungültig, bis das nächste Mal auf dieses Objekt zugegriffen wird.
- Beliebige Fehler bei der Berechtigung (SQLSTATE-Werte 42501, SQL0551N).

Ein Objekt kann erfolgreich erstellt werden, auch wenn durch seinen Hauptteil mehrere Fehler verursacht werden. Die Warnung, die daraufhin zurückgegeben wird, enthält den Namen des ersten nicht definierten, ungültigen oder nicht berechtigten Objekts, das bei der Kompilierung festgestellt wird. Die Katalogsicht SYSCAT.INVALIDOBJECTS enthält Informationen zu ungültigen Objekten.

Mit der Prozedur ADMIN_REVALIDATE_DB_OBJECTS können Sie vorhandene Objekte reaktivieren, die als inaktiv gekennzeichnet wurden.

Beispiel

```
create view v2 as select * from v1
```

Falls 'v1' nicht vorhanden ist, wird die Anweisung CREATE VIEW erfolgreich ausgeführt, jedoch bleibt 'v2' ungültig.

Option REPLACE in verschiedenen CREATE-Anweisungen

Die Klausel **OR REPLACE** in der Anweisung CREATE für verschiedene Objekte, wie zum Beispiel Aliasnamen, Funktionen, Module, Kurznamen, Prozeduren (einschließlich föderierten Prozeduren), Sequenzen, Trigger, Variablen und Sichten, bietet die Möglichkeit, das Objekt zu ersetzen, wenn es bereits vorhanden ist. Anderenfalls wird es erstellt. Dadurch verringert sich der Aufwand erheblich, der zum Ändern eines Datenbankschemas erforderlich ist.

Die Zugriffsrechte, die zuvor für ein Objekt erteilt wurden, bleiben erhalten, wenn dieses Objekt ersetzt wird. In anderer Hinsicht ist eine Anweisung CREATE OR REPLACE semantisch einer Anweisung DROP gefolgt von einer Anweisung CREATE ähnlich. Im Fall von Funktionen, Prozeduren und Triggern erstreckt sich die Unterstützung auf Inline-Objekte und kompilierte Objekte.

Im Fall von Funktionen und Prozeduren gilt die Unterstützung für SQL-Funktionen und externe Funktionen und Prozeduren. Wenn ein Modul ersetzt wird, werden alle Objekte im Modul gelöscht. Die neue Version des Moduls enthält keine Objekte.

Objekte, die von einem Objekt (direkt oder indirekt) abhängen, das ersetzt wird, werden inaktiviert. Die Reaktivierung aller abhängigen Objekte nach einer Ersetzungsoperation erfolgt immer unverzüglich nach der Inaktivierung, auch wenn der Datenbankkonfigurationsparameter **auto_reval** auf den Wert DISABLED gesetzt ist.

Beispiel

Ersetzen der Sicht 'v1', die abhängige Objekte hat:

```
create table t1 (c1 int, c2 int);
create table t2 (c1 int, c2 int);

create view v1 as select * from t1;
create view v2 as select * from v1;

create function foo1()
 language sql
 returns int
 return select c1 from v2;

create or replace v1 as select * from t2;

select * from v2;

values foo1();
```

Die ersetzte Version von 'v1' verweist nicht auf Tabelle 't1', sondern auf Tabelle 't2'. Sowohl 'v2' als auch 'foo1' werden durch die Anweisung CREATE OR REPLACE inaktiviert. Unter der Semantik der verzögerten Reaktivierung wird durch `select * from v2` die Sicht 'v2' erfolgreich reaktiviert, jedoch nicht die Funktion 'foo1', die durch 'values foo1()' reaktiviert wird. Unter der Semantik der unverzüglichen Reaktivierung werden sowohl 'v2' als auch 'foo1' durch die Anweisung CREATE OR REPLACE erfolgreich reaktiviert.

Kapitel 12. Tabellen

Bei Tabellen handelt es sich um logische Strukturen, die vom Datenbankmanager verwaltet werden. Tabellen bestehen aus Spalten und Zeilen.

Am Schnittpunkt jeder Spalte und jeder Zeile gibt es ein bestimmtes Datenelement, das *Wert* genannt wird. Eine *Spalte* ist eine Gruppe von Werten desselben Typs oder eines der Untertypen dieses Typs. Eine *Zeile* ist eine Folge von Werten, die so angeordnet sind, dass der *n*te Wert ein Wert der *n*ten Spalte der Tabelle ist.

Ein Anwendungsprogramm kann die Reihenfolge bestimmen, in der die Zeilen in die Tabelle eingefügt werden. Die tatsächliche Reihenfolge der Zeilen wird jedoch durch den Datenbankmanager bestimmt und kann in der Regel nicht gesteuert werden. Das mehrdimensionale Clustering (MDC) stellt eine gewisse Clustering-methode bereit, bestimmt jedoch nicht die tatsächliche Anordnung unter den Zeilen.

Typen von Tabellen

DB2-Datenbanken speichern Daten in Tabellen. Neben Tabellen, die zum Speichern persistenter Daten verwendet werden, gibt es auch Tabellen, die zur Darstellung von Ergebnissen dienen, sowie Übersichtstabellen und temporäre Tabellen. Ferner bieten Tabellen mit mehrdimensionalem Clustering (MDC) besondere Vorteile in einer Data-Warehouse-Umgebung.

Basistabellen
Diese Tabellentypen enthalten persistente Daten. Es gibt verschiedene Typen von Basistabellen:

Reguläre Tabellen
Reguläre Tabellen mit Indizes sind die "allgemein einsetzbaren" Tabellen.

MDC-Tabellen (MDC = mehrdimensionales Clustering)
Diese Tabellentypen werden als Tabellen implementiert, die physisch auf der Grundlage mehrerer Schlüssel bzw. Dimensionen gleichzeitig in Clustern angeordnet werden. MDC-Tabellen werden beim Data-Warehousing und in umfangreichen Datenbankumgebungen verwendet. Clusterindizes in regulären Tabellen unterstützen das eindimensionale Clustering von Daten. MDC-Tabellen bieten die Vorteile des Datenclusterings in mehr als einer Dimension. MDC-Tabellen stellen ein *garantiertes Clustering* innerhalb der zusammengesetzten Dimensionen sicher. Demgegenüber ist es zwar möglich, einen Clusterindex für reguläre Tabellen zu haben, und der Datenbankmanager versucht in diesem Fall, das Clustering aufrechtzuerhalten, jedoch lässt sich das Clustering nicht garantieren und nicht in der Regel im Lauf der Zeit ab. MDC-Tabellen können mit partitionierten Tabellen koexistieren und können selbst partitionierte Tabellen sein.

Tabellen mit mehrdimensionalem Clustering werden in einer DB2 pureScale-Umgebung nicht unterstützt.

ITC-Tabellen (ITC = Insert Time Clustering, Clustering anhand der Einfügungszeit)
Diese Tabellen ähneln vom Konzept her sowie physisch den MDC-

Tabellen. Zeilen werden jedoch nicht nach einer oder mehreren benutzerdefinierten Dimensionen, sondern nach dem Zeitpunkt in Gruppen zusammengefasst, an dem sie in die Tabelle eingefügt wurden. Bei ITC-Tabellen kann es sich um partitionierte Tabellen handeln.

ITC-Tabellen werden in einer DB2 pureScale-Umgebung nicht unterstützt.

Bereichsclustertabelle (RCT-Tabelle)
Diese Tabellentypen werden als sequenzielle Datencluster implementiert und bieten schnellen, direkten Zugriff. Jeder Datensatz in der Tabelle verfügt über eine vorbestimmte Satz-ID, die eine interne Kennung zur Lokalisierung eines Datensatzes in einer Tabelle darstellt. RCT-Tabellen werden verwendet, wenn die Daten eine dichte Clusterbildung über eine oder mehrere Tabellenspalten aufweisen. Die höchsten und niedrigsten Werte in den Spalten definieren den Bereich der möglichen Werte. Sie verwenden diese Spalten für den Zugriff auf Datensätze in der Tabelle. Dies ist die optimale Methode zur Nutzung des Merkmals der vorbestimmten Satz-IDs (RIDs) von RCT-Tabellen.

Bereichsclustertabellen werden in einer DB2 pureScale-Umgebung nicht unterstützt.

Temporale Tabellen
Diese Tabellentypen werden zum Zuordnen von zeitbasierten Statusinformationen zu Ihren Daten verwendet. Die Daten in Tabellen ohne temporale Unterstützung sind aktuell (d. h. zum gegenwärtigen Zeitpunkt gültig). Im Unterschied dazu können Daten in temporalen Tabellen für einen Zeitraum gültig sein, der vom Datenbanksystem und/oder von Benutzeranwendungen definiert wird. Beispielsweise kann in einer Datenbank das Verlaufsprotokoll für eine Tabelle gespeichert sein (d. h. die Inhalte gelöschter Zeilen oder die ursprünglichen Werte für Zeilen, die aktualisiert wurden), damit Sie frühere Stände Ihrer Daten abrufen können. Außerdem können Sie einer Datenzeile einen Datumsbereich zuordnen, um festzulegen, wann die Daten von Ihren Anwendungen oder Geschäftsregeln als gültig angesehen werden.

Temporäre Tabellen
Diese Tabellentypen werden als temporäre Arbeitstabellen für eine ganze Reihe von Datenbankoperationen verwendet. *Deklarierte temporäre Tabellen* (DGTTs) werden im Systemkatalog nicht aufgeführt. Dadurch sind sie nicht persistent und können von anderen Anwendungen nicht verwendet bzw. nicht mit anderen Anwendungen gemeinsam genutzt werden. Wenn die Anwendung, die diese Tabelle verwendet, beendet wird oder die Verbindung zur Datenbank trennt, werden alle Daten in dieser Tabelle und die Tabelle selbst gelöscht. Im Gegensatz dazu werden *erstellte temporäre Tabellen* (CGTTs) im Systemkatalog aufgeführt und müssen nicht in jeder Sitzung, in der sie verwendet werden, definiert werden. Infolgedessen sind sie persistent und können mit anderen Anwendungen über verschiedene Verbindungen hinweg gemeinsam genutzt werden.

Keiner der Typen von temporären Tabellen unterstützt folgende Elemente:

- Spalten mit einem benutzerdefinierten Verweistyp oder benutzerdefinierten strukturierten Typ
- LONG VARCHAR-Spalten

Darüber hinaus können XML-Spalten nicht in erstellten temporären Tabellen verwendet werden.

Materialized Query Tables (MQTs)
Diese Tabellentypen werden durch eine Abfrage definiert, mit der auch die Daten für die Tabelle ermittelt werden. Durch MQTs kann die Leistungsfähigkeit von Abfragen erhöht werden. Wenn der Datenbankmanager erkennt, dass ein Teil einer Abfrage durch eine Übersichtstabelle aufgelöst werden kann, kann der Datenbankmanager die Abfrage so abwandeln, dass die entsprechende Übersichtstabelle verwendet wird. Diese Entscheidung basiert auf Datenbankkonfigurationseinstellungen wie z. B. den Sonderregistern CURRENT REFRESH AGE und CURRENT QUERY OPTIMIZATION. Eine Übersichtstabelle ist ein spezialisierter Typ von MQT.

Sie können alle der oben genannten Typen von Tabellen mit der Anweisung CREATE TABLE erstellen.

Abhängig von der Art Ihrer Daten bietet ein Tabellentyp möglicherweise eine bestimmte Funktionalität, durch die sich die Speicher- und Abfrageleistung optimieren lassen. Wenn Sie z. B. mit Datensätzen arbeiten, die über eine lockere Clusterbildung (ohne monotones Wachstum) verfügen, sollten Sie prüfen, ob eine reguläre Tabelle und Indizes eingesetzt werden können. Wenn Sie über Datensätze verfügen, die doppelte (jedoch nicht eindeutige) Werte im Schlüssel aufweisen, sollten Sie keine Bereichsclustertabelle verwenden. Auch wenn Sie vorab keine festgelegte Menge an Plattenspeicherplatz für Bereichsclustertabellen zuordnen können, sollten Sie diesen Tabellentyp nicht verwenden. Wenn Sie Daten haben, die das Potenzial für ein Clustering nach mehreren Dimensionen haben, wie zum Beispiel eine Tabelle, in der Einzelhandelsumsätze nach geografischer Region, Unternehmensbereich und Lieferant aufgezeichnet werden, könnte sich eine Tabelle mit mehrdimensionalem Clustering (MDC-Tabelle) für diese Zwecke anbieten.

Neben den verschiedenen, oben beschriebenen Tabellentypen stehen Ihnen auch Optionen für solche Merkmale für *Partitionierung*, durch die sich die Leistung für Tasks wie das Einlagern definierter Datenbestände (Rollin) verbessern lassen. Partitionierte Tabellen können darüber hinaus auch wesentlich mehr Informationen als reguläre, nicht partitionierte Tabellen aufnehmen. Außerdem können Sie auch Funktionen wie die *Komprimierung* nutzen, mit deren Hilfe Sie Ihre Datenspeicherkosten erheblich senken können.

Entwerfen von Tabellen

Beim Entwerfen von Tabellen müssen Sie mit bestimmten Konzepten vertraut sein, den Platzbedarf für Tabellen- und Benutzerdaten bestimmen sowie festlegen, ob bestimmte Funktionen, wie zum Beispiel die Komprimierung und das optimistische Sperren, genutzt werden sollen.

Beim Entwerfen von partitionierten Tabellen müssen Sie sich mit den Partitionierungskonzepten vertraut machen, wie zum Beispiel:

- Datenorganisationsschemata
- Tabellenpartitionierungsschlüssel
- Für die Verteilung von Daten auf Datenpartitionen verwendete Schlüssel
- Für MDC-Dimensionen verwendete Schlüssel

Informationen zu diesen und anderen Partitionierungskonzepten finden Sie in „Tabellenpartitionierung und Datenorganisationsschemata” auf Seite 367.

Entwurfskonzepte für Tabellen

Beim Entwerfen von Tabellen müssen Sie sich mit einigen zugehörigen Konzepten vertraut machen.

Datentypen und Tabellenspalten

Wenn Sie eine Tabelle erstellen, müssen Sie angeben, welcher Typ von Daten in den einzelnen Spalten gespeichert werden soll. Durch sorgfältiges Berücksichtigen der Spezifik der Daten, die verwaltet werden sollen, können Sie Ihre Tabellen auf eine Art und Weise einrichten, die eine optimale Abfrageleistung ermöglicht, den physischen Speicherbedarf minimiert und die spezialisierten Funktionen zur Bearbeitung der verschiedenen Arten von Daten, wie zum Beispiel arithmetische Operationen für numerische Daten oder Vergleichsoperationen für Datums- und Zeitwerte, bereitstellt.

In Abb. 34 sind die Datentypen aufgeführt, die von DB2-Datenbanken unterstützt werden.
Wenn Sie Ihre Datenbankspalten deklarieren, stehen Ihnen alle diese Datentypen

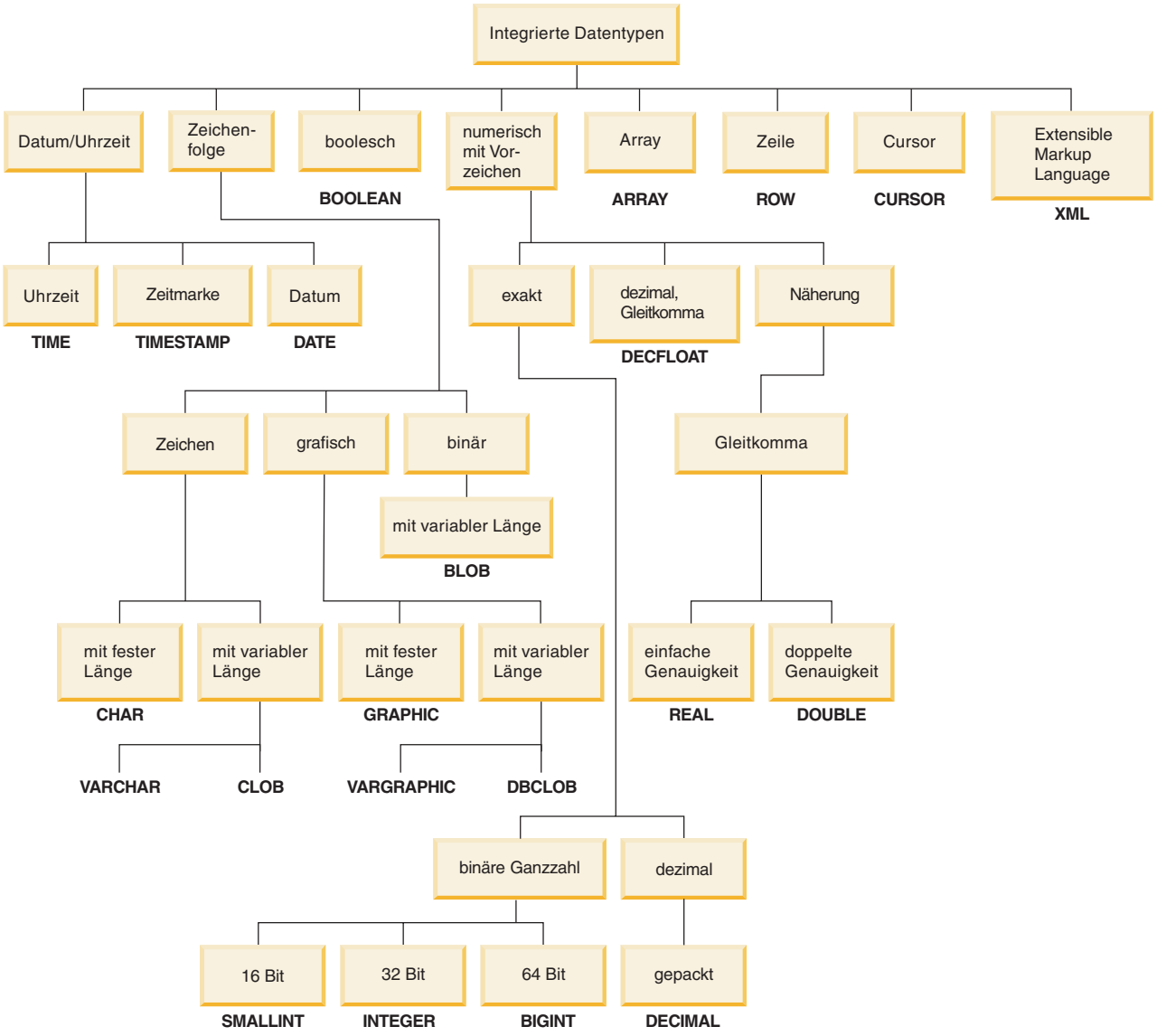


Abbildung 34. Integrierte Datentypen

zur Auswahl. Neben den integrierten Datentypen können Sie außerdem eigene *benutzerdefinierte* Datentypen erstellen, die auf den integrierten Typen basieren. Sie könnten zum Beispiel einen Mitarbeiter durch Attribute für Name, Jobbezeichnung, Jobebene, Einstellungsdatum und Gehalt in einem benutzerdefinierten *strukturierten* Typ darstellen, der Daten der Typen VARCHAR (Name, Jobbezeichnung), SMALLINT (Jobebene), DATE (Einstellungsdatum) und DECIMAL (Gehalt) umfasst.

Generierte Spalten

Eine generierte Spalte wird in einer Tabelle definiert, bei der der gespeicherte Wert mithilfe eines Ausdrucks ermittelt wird, anstatt durch eine Einfüge- oder Aktualisierungsoperation angegeben zu werden.

Beim Erstellen einer Tabelle, bei der bestimmte Ausdrücke und Vergleichselemente ständig verwendet werden, können eine oder mehrere generierte Tabellen zu dieser Tabelle hinzugefügt werden. Durch die Verwendung einer generierten Spalte ergibt sich die Möglichkeit, bei der Durchführung von Abfragen in den Tabellendaten eine verbesserte Leistung zu erzielen.

Es gibt zum Beispiel zwei Fälle, in denen die Auswertung von Ausdrücken aufwendig sein kann, wenn die Leistung wichtig ist:

1. Die Auswertung des Ausdrucks muss während einer Abfrage häufig durchgeführt werden.
2. Die Berechnung ist komplex.

Um die Leistung einer Abfrage zu verbessern, können Sie eine zusätzliche Spalte definieren, die die Resultate des Ausdrucks enthält. Beim Absetzen einer Abfrage mit demselben Ausdruck kann die generierte Spalte dann direkt verwendet werden. Andernfalls kann die Komponente zum Umschreiben von Abfragen, die im Optimierungsprogramm implementiert ist, den Ausdruck gegen die generierte Spalte austauschen.

Wenn bei Abfragen Daten von zwei oder mehr Tabellen verknüpft werden, ermöglicht das Hinzufügen einer generierten Spalte dem Optimierungsprogramm die Auswahl potenziell besserer Joinstrategien.

Generierte Spalten können zur Verbesserung der Leistung von Abfragen eingesetzt werden. Aus diesem Grund werden generierte Spalten häufig nach dem Erstellen und Füllen einer Tabelle mit Werten hinzugefügt.

Beispiele

Das folgende Beispiel zeigt die Definition einer generierten Spalte in der Anweisung CREATE TABLE:

```
CREATE TABLE t1 (c1 INT,
                 c2 DOUBLE,
                 c3 DOUBLE GENERATED ALWAYS AS (c1 + c2)
                 c4 GENERATED ALWAYS AS
                    (CASE WHEN c1 > c2 THEN 1 ELSE NULL END))
```

Nach dem Erstellen dieser Tabelle können mithilfe der generierten Spalten Indizes erstellt werden. Beispiel:

```
CREATE INDEX i1 ON t1(c4)
```

Abfragen können die Vorteile generierter Spalten nutzen. Beispiel:

```
SELECT COUNT(*) FROM t1 WHERE c1 > c2
```

Diese Abfrage könnte folgendermaßen geschrieben werden:

```
SELECT COUNT(*) FROM t1 WHERE c4 IS NOT NULL
```

Weiteres Beispiel:

```
SELECT c1 + c2 FROM t1 WHERE (c1 + c2) * c1 > 100
```

Diese Abfrage könnte folgendermaßen geschrieben werden:

```
SELECT c3 FROM t1 WHERE c3 * c1 > 100
```

Verdeckte Spalten

Wenn eine Tabellenspalte mit dem Attribut IMPLICITLY HIDDEN definiert ist, steht diese Spalte nur dann zur Verfügung, wenn explizit auf sie verwiesen wird. Wenn beispielsweise eine Abfrage SELECT * für eine Tabelle ausgeführt wird, werden implizit verdeckte Spalten in der Ergebnistabelle nicht zurückgegeben. Auf eine implizit verdeckte Spalte kann immer explizit verwiesen werden, wenn ein Spaltenname angegeben werden kann.

In Fällen, in denen Spalten und die zugehörigen Einträge vom Datenbankmanager generiert werden, kann das Definieren dieser Spalten als implizit verdeckt (IMPLICITLY HIDDEN) potenziell negative Auswirkungen auf die Anwendungen minimieren. Beispiel: Eine temporale Tabelle für Systemzeitraum verfügt über drei Spalten, deren Werte vom Datenbankmanager generiert werden. Der Datenbankmanager verwendet diese Spalten zur Archivierung von Langzeitversionen der einzelnen Tabellenzeilen. Die meisten Geschäftsanwendungen verarbeiten zwar Protokolldaten, würden diese drei generierten Spalten aber nicht verwenden. Das Verdecken dieser Spalten vor den Anwendungen kann zur Reduzierung der Verarbeitungszeit für Anwendungen beitragen.

Beim Einfügen von Daten in eine Tabelle erwartet eine Anweisung INSERT ohne Spaltenliste keine Werte für implizit verdeckte Spalten. Wenn in diesen Fällen die Eingabe einen Wert für eine implizit verdeckte Spalte enthält, weist dieser Wert keine entsprechende Zielspalte auf und es wird ein Fehler zurückgegeben (SQLSTATE 42802). Da eine Anweisung INSERT ohne Spaltenliste keine Werte für implizit verdeckte Spalten enthält, müssen alle Spalten, die als implizit verdeckt definiert sind und den Wert NOT NULL aufweisen, einen definierten Standardwert haben.

Beim Füllen einer Tabelle mit Daten aus einer Eingabedatei erfordern Dienstprogramme wie IMPORT, INGEST und LOAD, dass Sie angeben, ob Daten für die verdeckten Spalten in die Operation eingeschlossen sind. Wenn keine Spaltenliste angegeben ist, müssen Dienstprogramme für das Versetzen von Daten die Dateitypänderungswerte *implicitlyhiddeninclude* oder *implicitlyhiddenmissing* verwenden, wenn mit Tabellen gearbeitet wird, die implizit verdeckte Spalten enthalten. Sie können auch die Registrierdatenbankvariable DB2_DMU_DEFAULT verwenden, um das Standardverhalten für den Fall festzulegen, wenn Dienstprogramme für das Versetzen von Daten auf Tabellen treffen, die implizit verdeckte Spalten enthalten. In gleicher Weise erfordert EXPORT, dass Sie angeben, ob Daten für die verdeckten Spalten in die Operation eingeschlossen sind.

Das Attribut IMPLICITLY HIDDEN kann für eine Tabellenspalte mit der Anweisung CREATE TABLE für neue Tabellen oder mit der Anweisung ALTER TABLE für vorhandene Tabellen definiert werden. Wenn eine Tabelle unter Verwendung der Anweisung CREATE TABLE mit der Klausel LIKE erstellt wird, werden alle implizit verdeckte Spalten in der Quellentabelle von der neuen Tabelle übernommen. Mit der Anweisung ALTER TABLE können verdeckte Spalten in nicht verdeckte Spalten (und umgekehrt) geändert werden. Wenn eine Tabelle geändert wird, um das HIDDEN-Attribut einiger Spalten zu ändern, kann dies Einfluss auf das Verhalten von Dienstprogrammen für das Versetzen von Daten haben, die mit der Tabelle arbeiten. Dies kann z. B. bedeuten, dass eine Ladeoperation, die vor der zum Definieren verdeckter Spalten durchgeführten Tabellenänderung erfolgreich ausgeführt wurde, nun einen Fehler zurückgibt (SQLCODE SQL2437N).

Die Liste mit den Namen für die Spalten einer Ergebnistabelle aus einer SELECT-Abfrage, die mit der Option *exponierter_name.** ausgeführt wird, enthält keine im-

plizit verdeckten Spalten. Eine mit order-by-clause ausgeführte SELECT-Abfrage kann unter *einfacher_spaltenname* implizit verdeckte Spalten enthalten.

Wenn in einer MQT-Definition (MQT - Materialized Query Table, gespeicherte Abfragetabelle) explizit auf eine implizit verdeckte Spalte verwiesen wird, ist diese Spalte Teil der MQT. Das Attribut IMPLICITLY HIDDEN wird jedoch nicht von der Spalte in der MQT übernommen. Dieses Verhalten kommt auch bei Sichten und Tabellen zur Anwendung, die mit der Klausel 'as-result-table' erstellt werden.

Auf eine implizit verdeckte Spalte kann in einer Anweisung CREATE INDEX, einer Anweisung ALTER TABLE oder in einer referenziellen Integritätsbedingung explizit verwiesen werden.

Für jede als implizit verdeckt definierte Spalte ist eine Übergangsvariable vorhanden. Auf eine Übergangsvariable, die einer implizit verdeckten Spalte entspricht, kann im Hauptteil eines Triggers verwiesen werden.

Implizit verdeckte Spalten werden in erstellten temporären Tabellen und deklarierten temporären Tabellen nicht unterstützt.

Die verdeckten Spalten für eine Tabelle können mit dem Befehl DESCRIBE angezeigt werden.

```
DESCRIBE TABLE tabellenname SHOW DETAIL
```

Beispiel

- *Beispiel 1:* In der folgenden Anweisung wird eine Spalte mit einer implizit verdeckten Spalte erstellt.

  ```
  CREATE TABLE CUSTOMER
  (
   CUSTOMERNO       INTEGER NOT NULL,
   CUSTOMERNAME     VARCHAR(80),
   PHONENO          CHAR(8) IMPLICITLY HIDDEN
  );
  ```

  SELECT * gibt nur die Spalteneinträge für CUSTOMERNO und CUSTOMERNAME zurück. Beispiel:

  ```
  A123, ACME
  B567, First Choice
  C345, National Chain
  ```

  Die Einträge der Spalte PHONENO sind verdeckt, sofern nicht explizit auf sie verwiesen wird.

  ```
  SELECT CUSTOMERNO, CUSTOMERNAME, PHONENO
     FROM CUSTOMER
  ```

- *Beispiel 2:* Wenn eine Datenbank implizit verdeckte Spalten enthält, müssen Sie angeben, ob Daten für die verdeckten Spalten in Operationen zum Versetzen von Daten eingeschlossen sind. Im folgenden Beispiel wird LOAD verwendet, um die unterschiedlichen Methoden zu zeigen, mit denen angegeben werden kann, ob Daten für verdeckte Spalten eingeschlossen sind:
  - Verwenden Sie die *einfügespalte*, um explizit die Spalten anzugeben, in die Daten eingefügt werden sollen.

    ```
    db2 load from delfile1 of del
       insert into table1 (c1, c2, c3,...)
    ```

– Verwenden Sie einen der Dateitypänderungswerte für verdeckte Spalten: Geben Sie **implicitlyhiddeninclude** an, wenn die Eingabedatei Daten für die verdeckten Spalten enthält, oder **implicitlyhiddenmissing**, wenn die Eingabedatei solche Daten nicht enthält.

```
db2 load from delfile1 of del modified by implicitlyhiddeninclude
   insert into table1
```

– Verwenden Sie die Registrierdatenbankvariable DB2_DMU_DEFAULT auf der Serverseite, um das Verhalten für den Fall festzulegen, wenn Dienstprogramme für das Versetzen von Daten auf Tabellen treffen, die implizit verdeckte Spalten enthalten.

```
db2set DB2_DMU_DEFAULT=IMPLICITLYHIDDENINCLUDE
db2 load from delfile1 of del insert into table1
```

Automatische Nummerierung und ID-Spalten

Eine Identitätsspalte ermöglicht DB2 das automatische Generieren eines eindeutigen numerischen Wertes für jede Zeile, die der Tabelle hinzugefügt wird.

Beim Erstellen einer Tabelle, wobei Sie jede der Tabelle hinzugefügte Zeile eindeutig identifizieren müssen, können Sie der Tabelle eine Identitätsspalte hinzufügen. Um einen eindeutigen numerischen Wert für jede Zeile sicherzustellen, die der Tabelle hinzugefügt wird, sollten Sie einen eindeutigen Index für die Identitätsspalte definieren oder die Spalte als Primärschlüssel deklarieren.

Außerdem können Identitätsspalten für die Zuordnung von Bestellnummern, Personalnummern, Produktnummern oder Ereignisnummern verwendet werden. Die Werte für eine Identitätsspalte können mit dem DB2-Datenbankmanager unter Verwendung von ALWAYS oder BY DEFAULT generiert werden.

Eine mit GENERATED ALWAYS definierte Identitätsspalte erhält immer vom DB2-Datenbankmanager generierte Werte. Für Anwendungen ist die Angabe eines expliziten Wertes nicht zulässig. Eine mit GENERATED BY DEFAULT definierte Identitätsspalte ermöglicht Anwendungen die explizite Angabe eines Wertes für die Identitätsspalte. Wenn die Anwendung keinen Wert bereitstellt, wird ein entsprechender Wert von DB2 generiert. Da der Wert von der Anwendung gesteuert wird, kann seine Eindeutigkeit durch DB2 nicht garantiert werden. Die Klausel GENERATED BY DEFAULT sollte für die Datenweitergabe eingesetzt werden, wenn der Inhalt einer vorhandenen Tabelle kopiert werden soll. Andere Einsatzbereiche sind das Entladen und das erneute Laden einer Tabelle.

Nach der Erstellung müssen Sie als Erstes die Spalte mit der Option DEFAULT hinzufügen, um den vorhandenen Standardwert zu erhalten. Anschließend können Sie den Standardwert ändern, sodass die Spalte zu einer einer Identitätsspalte werden kann.

Wenn Zeilen in eine Tabelle mit der Angabe expliziter Werte für Identitätsspalten eingefügt werden, wird der nächste intern generierte Wert nicht aktualisiert und kann Konflikte mit anderen in der Tabelle vorhandenen Werten verursachen. Doppelte Werte führen zu einer Fehlernachricht, wenn die Eindeutigkeit der Werte in der Identitätsspalte durch einen Primärschlüssel oder einen für die Identitätsspalte definierten eindeutigen Index überprüft wird.

Wenn Sie eine Identitätsspalte in einer neuen Tabelle definieren möchten, verwenden Sie in der Anweisung CREATE TABLE die Klausel AS IDENTITY.

Beispiel

Das folgende Beispiel zeigt die Definition einer Identitätsspalte in der Anweisung CREATE TABLE:

```
CREATE TABLE table (col1 INT,
                    col2 DOUBLE,
                    col3 INT NOT NULL GENERATED ALWAYS AS IDENTITY
                                      (START WITH 100, INCREMENT BY 5))
```

In diesem Beispiel wurde die dritte Spalte als Identitätsspalte definiert. Sie können darüber hinaus den in der Spalte verwendeten Wert zum eindeutigen Identifizieren aller hinzugefügten Zeilen verwenden. Im vorliegenden Beispiel wird für die erste eingegebene Zeile der Wert „100" in der Spalte platziert. Für jede nachfolgende, der Tabelle hinzugefügte Zeile wird der zugehörige Wert um 5 erhöht.

Einschränken von Spaltendaten durch Integritätsbedingungen, Standardwerte und Nulleinstellungen

Daten müssen häufig bestimmten Einschränkungen oder Regeln genügen. Solche Einschränkungen können für Einzelinformationen, wie zum Beispiel das Format und die Reihenfolge numerischer Werte, oder für mehrere Informationen gelten.

Informationen zu diesem Vorgang

Optionalität der Spaltendateneingabe

Nullwerte stellen unbekannte Status dar. Standardmäßig unterstützen alle integrierten Datentypen das Vorhandensein von Nullwerten. Es ist jedoch möglich, dass Geschäftsregeln vorgeben, dass für einige Spalten in jedem Fall ein Wert angegeben werden muss, wie zum Beispiel Notfallinformationen. Zur Definition einer solchen Bedingung können Sie die Integritätsbedingung NOT NULL angeben, die sicherstellt, dass eine bestimmte Spalte einer Tabelle nie einen Nullwert aufnimmt. Wenn die Integritätsbedingung NOT NULL für eine bestimmte Spalte definiert wurde, schlägt jede Einfüge- oder Aktualisierungsoperation fehl, die versucht, in dieser Spalte einen Nullwert zu speichern.

Standardspaltendatenwerte

Ebenso wie Geschäftsregeln vorschreiben können, dass immer ein Wert anzugeben ist, können andere Geschäftsregeln vorschreiben, was dieser Wert zu sein hat, zum Beispiel dass der Wert für das Geschlecht eines Mitarbeiters entweder M oder F sein muss. Die Standardwertintegritätsbedingung für eine Spalte dient zur Sicherstellung, dass eine bestimmte Spalte einer Tabelle immer einen vordefinierten Wert erhält, wenn eine Zeile, die keinen bestimmten Wert für diese Spalte enthält, in die Tabelle eingefügt wird. Der für eine Spalte vorgesehene Standardwert kann null sein, kann ein Integritätswert sein, der mit dem Datentyp der Spalte kompatibel ist, oder kann ein Wert sein, der vom Datenbankmanager bereitgestellt wird. Weitere Informationen finden Sie in „Standardspalten- und Datentypdefinitionen" auf Seite 329.

Schlüssel

Ein Schlüssel ist eine einzelne Spalte oder eine Gruppe von Spalten in einer Tabelle oder einem Index, die zur Ermittlung einer bestimmten Zeile bzw. für den Zugriff auf diese Zeile verwendet werden kann. Jede Spalte kann Teil eines Schlüssels sein, und dieselbe Spalte kann Teil mehrerer Schlüssel sein. Ein Schlüssel, der aus einer einzelnen Spalte besteht, wird als Atomarschlüssel (engl. atomic key) bezeichnet. Ein Schlüssel, der sich aus mehreren Spalten zusammensetzt, wird als zusammengesetzter Schlüssel bezeichnet. Neben ihrer Eigenschaft als atomare oder zusammengesetz-

te Schlüssel werden Schlüssel auch danach klassifiziert, wie sie zur Implementierung von Integritätsbedingungen eingesetzt werden:

- Ein eindeutiger Schlüssel wird zur Implementierung eindeutiger Integritätsbedingungen verwendet.
- Ein Primärschlüssel wird zur Implementierung von Entitätenintegritätsbedingungen verwendet. (Ein Primärschlüssel ist ein spezieller Typ von eindeutigem Schlüssel, der keine Nullwerte unterstützt.)
- Ein Fremdschlüssel wird zur Implementierung referenzieller Integritätsbedingungen verwendet. (Fremdschlüssel müssen auf Primärschlüssel oder eindeutige Schlüssel verweisen. Fremdschlüssel haben keine zugehörigen Indizes.)

Schlüssel werden in der Regel bei der Deklaration einer Tabelle, eines Index oder einer Definition einer referenziellen Integritätsbedingung angegeben.

Integritätsbedingungen

Integritätsbedingungen sind Regeln, die die Werte in einer Tabelle einschränken, die eingefügt, gelöscht oder aktualisiert werden können. Es gibt Prüfungen auf Integritätsbedingungen, Integritätsbedingungen über Primärschlüssel, referenzielle Integritätsbedingungen, eindeutige Integritätsbedingungen, Integritätsbedingungen über eindeutige Schlüssel, Integritätsbedingungen über Fremdschlüssel und informative Integrationsbedingungen. Detaillierte Informationen zu den einzelnen Typen von Integritätsbedingungen finden Sie in Kapitel 13, „Integritätsbedingungen", auf Seite 447 bzw. „Typen von Integritätsbedingungen" auf Seite 447.

Standardspalten- und Datentypdefinitionen:

Bestimmte Spalten- und Datentypen verfügen über vordefinierte oder zugeordnete Standardwerte.

Die folgenden Standardspaltenwerte für die verschiedenen Datentypen sind möglich:

- *NULL*
- *0:* Wird für ganze Zahlen ohne erweiterte Genauigkeit, ganze Zahlen, Dezimalzahlen, Gleitkommazahlen mit einfacher Genauigkeit, Gleitkommazahlen mit doppelter Genauigkeit und Dezimalgleitkommazahlen verwendet.
- *Leerzeichen:* Wird für Zeichenfolgen mit fester Länge und für Doppelbytezeichenfolgen mit fester Länge verwendet.
- *Zeichenfolgewert der Länge null:* Wird für Zeichenfolgen mit variabler Länge, große Binärobjekte, große Zeichenobjekte und große Doppelbytezeichenobjekte verwendet.
- *Datum:* Dabei handelt es sich um das Systemdatum, an dem die Zeile eingefügt wurde (abgerufen aus dem Sonderregister CURRENT_DATE). Wenn einer vorhandenen Tabelle eine Datumsspalte hinzugefügt wird, wird vorhandenen Zeilen das Datum January, 01, 0001 zugeordnet.
- *Zeit oder Zeitmarke:* Dabei handelt es sich um die Systemzeit bzw. um das Systemdatum/die Systemzeit, an dem/zu der die Anweisung eingefügt wurde (abgerufen aus dem Sonderregister CURRENT_TIME). Wenn einer vorhandenen Tabelle eine Zeitspalte hinzugefügt wird, wird vorhandenen Zeilen die Uhrzeit 00:00:00 bzw. eine Zeitmarke zugeordnet, die das Datum January, 01, 0001 und die Uhrzeit 00:00:00 enthält.

Anmerkung: Alle Zeilen erhalten denselben Standardwert für die Zeit/Zeitmarke für eine angegebene Anweisung.

- *Einzigartiger benutzerdefinierter Datentyp:* Dabei handelt es sich um den integrierten Standardwert für den Basisdatentyp des einzigartigen benutzerdefinierten Datentyps (Umsetzung in den einzigartigen benutzerdefinierten Datentyp).

Ordnen von Spalten zur Minimierung der Aktualisierungsprotokollierung:

Wenn Sie Spalten mit der Anweisung CREATE TABLE definieren, müssen Sie die Reihenfolge der Spalten insbesondere bei aktualisierungsintensiven Auslastungen berücksichtigen. Spalten, die häufig aktualisiert werden, sollten gruppiert und näher zum Ende bzw. am Ende der Tabellendefinition definiert werden. Dies verbessert die Leistung, verringert die Anzahl der zu protokollierenden Byte und senkt die Anzahl der zu schreibenden Protokollseiten. Darüber hinaus vermindert diese Vorgehensweise den Speicherplatzbedarf für aktive Protokolle bei Transaktionen, die eine große Anzahl von Aktualisierungen ausführen.

Der Datenbankmanager nimmt nicht automatisch an, dass Spalten, die in der SET-Klausel einer UPDATE-Anweisung angegeben sind, hinsichtlich des Werts geändert werden sollen. Zur Begrenzung des Aufwands der Indexpflege und des zu protokollierenden Umfangs der Zeile vergleicht die Datenbank den neuen Spaltenwert mit dem alten Spaltenwert, um festzustellen, ob die Spalte geändert wird. Nur Spalten, deren Wert sich ändert, werden als Spalten behandelt, die aktualisiert werden. Ausnahmen von diesem UPDATE-Verhalten ergeben sich für Spalten, bei denen die Daten außerhalb der Datenzeile gespeichert werden (Spaltentypen LONG, LOB, ADT und XML), oder für Spalten mit fester Länge, wenn die Registrierdatenbankvariable DB2ASSUMEUPDATE aktiviert ist. In diesen Ausnahmefällen wird angenommen, dass sich der Spaltenwert ändert, sodass kein Vergleich zwischen dem neuen und dem alten Spaltenwert ausgeführt wird.

Es gibt vier unterschiedliche Typen von UPDATE-Protokollsätzen.

- Vollständige Protokollierung der Vor- und Nachzeilenimages. Das gesamte Vorimage und Nachimage der Zeile wird protokolliert. Dies ist der einzige Typ von Protokollierung, der für Tabellen mit aktivierter Option DATA CAPTURE CHANGES ausgeführt wird, und hat die höchste Anzahl von Byte zur Folge, die für eine Aktualisierung an einer Zeile protokolliert wird.
- Protokollierung des vollständigen Vorzeilenimage, der geänderten Byte und für Aktualisierungen, die die Zeile vergrößern, wobei neue Daten an das Ende der Zeile angehängt werden. Diese Protokollierung erfolgt für Datenbanken, die die Funktion zur Verarbeitung nur der zurzeit festgeschriebenen Daten unterstützen, wenn die Option DATA CAPTURE CHANGES für die Tabelle nicht aktiviert ist, wenn die Aktualisierung die erste Aktion an dieser Zeile für eine Transaktion ist. Dabei werden das Vorimage, das für die Funktion zur Verarbeitung nur der zurzeit festgeschriebenen Daten erforderlich ist, und das Minimum an zusätzlichen Daten protokolliert, das zur Ausführung von Wiederholungen und Rückgängigmachungen benötigt wird. Wenn häufig aktualisierte Spalten am Ende angeordnet werden, minimiert dies die Protokollierung für den geänderten Teil der Zeile.
- Vollständige XOR-Protokollierung. Die XOR-Unterschiede zwischen dem Vorimage und dem Nachimage der Zeile, angefangen vom ersten Byte, das sich ändert, bis zum Ende der kleineren Zeile sowie alle restlichen Byte in der längeren Zeile, werden protokolliert. Dies führt dazu, dass weniger Byte protokolliert werden als bei der Protokollierung der vollständigen Vor- und Nachimages, wobei die Anzahl von Datenbyte, die über die Kopfinformationen des Protokollsatzes hinausgehen, die Größe des größten Zeilenimages ist.

- Partielle XOR-Protokollierung. Die XOR-Unterschiede zwischen dem Vorimage und dem Nachimage der Zeile, angefangen vom ersten Byte, das sich ändert, bis zum letzten Byte, das sich ändert, werden protokolliert. Bytepositionen können die ersten oder die letzten Byte einer Spalte sein. Bei diesem Verfahren wird die geringste Anzahl von Byte protokolliert und der effizienteste Typ von Protokollsatz für eine Aktualisierung an einer Zeile generiert.

Wenn für die ersten beiden oben aufgeführten Typen von UPDATE-Protokollsätzen die Option DATA CAPTURE CHANGES für die Tabelle nicht aktiviert ist, hängt der Umfang der Daten, die für eine Aktualisierung protokolliert werden, von folgenden Faktoren ab:

- Die räumliche Nähe der aktualisierten Spalten (COLNO)
- Ob die aktualisierten Spalten eine feste oder variable Länge haben
- Ob die Zeilenkomprimierung (COMPRESS YES) aktiviert ist

Wenn sich die Gesamtlänge der Zeile auch bei aktivierter Zeilenkomprimierung nicht ändert, berechnet und schreibt der Datenbankmanager den optimalen partiellen XOR-Protokollsatz.

Wenn sich die Gesamtlänge der Zeile ändert, was bei der Aktualisierung von Spalten mit variabler Länge und bei aktivierter Zeilenkomprimierung der Regelfall ist, stellt der Datenbankmanager fest, welches Byte zuerst zu ändern ist und schreibt einen vollständigen XOR-Protokollsatz.

Integritätsbedingungen über Primärschlüssel, referenzielle Integritätsbedingungen, Prüfungen auf Integritätsbedingungen und eindeutige Integritätsbedingungen

Integritätsbedingungen sind Regeln, die die Werte begrenzen, die in eine Tabelle eingefügt, aus einer Tabelle gelöscht oder in einer Tabelle aktualisiert werden können.

Integritätsbedingungen über Primärschlüssel
: Eine Integritätsbedingung über Primärschlüssel ist eine Spalte oder eine Kombination von Spalten, die die gleichen Eigenschaften wie eine eindeutige Integritätsbedingung besitzt. Ein Primärschlüssel und Integritätsbedingungen über Fremdschlüssel dienen zur Definition von Beziehungen zwischen Tabellen.

Referenzielle Integritätsbedingungen (oder Integritätsbedingungen über Fremdschlüssel)
: Eine Integritätsbedingung über Fremdschlüssel (auch als referenzielle Integritätsbedingung bezeichnet) ist eine logische Regel über Werte in einer oder mehreren Spalten in mindestens einer Tabelle. Zum Beispiel enthält eine Gruppe von Tabellen gemeinsame Informationen über die Lieferanten einer Firma. Gelegentlich ändert sich der Name eines Lieferanten. Sie können eine referenzielle Integritätsbedingung definieren, die besagt, dass die Kennung (ID) des Lieferanten in einer Tabelle mit einer Lieferanten-ID in den Lieferanteninformationen übereinstimmen muss. Diese Integritätsbedingung verhindert Einfüge-, Aktualisierungs- oder Löschoperationen, die ansonsten zu fehlenden Lieferanteninformationen führen würden.

Prüfungen auf Integritätsbedingungen
: Eine Prüfung auf Integritätsbedingung (in Tabellen) definiert Einschränkungen für Daten, die einer bestimmten Tabelle hinzugefügt werden.

Eindeutige Integritätsbedingungen
: Eine eindeutige Integritätsbedingung (auch als Integritätsbedingung über

eindeutige Schlüssel bezeichnet) ist eine Regel, die untersagt, dass in einer oder mehreren Spalten innerhalb einer Tabelle doppelte Werte auftreten. Eindeutige Integritätsbedingungen werden in Form von eindeutigen Schlüsseln und Primärschlüsseln unterstützt.

Unicode-Tabellen und -Daten - Hinweise

Der Unicode-Zeichencodierungsstandard ist ein Codierungsschema für Zeichen mit fester Länge, das Zeichen fast aller lebenden Sprachen der Welt umfasst.

Weitere Informationen zu den Aspekten bezüglich Unicode-Tabellen und -Daten finden Sie in den folgenden Abschnitten:

- „Unicode-Zeichencodierung" in *Globalisierung*
- „Zeichenvergleiche auf der Basis von Sortierfolgen" in *Globalisierung*
- „Datums- und Zeitformate nach Gebietscodes" in *Globalisierung*
- „Konvertierungstabellendateien für Euro-fähige Codepages" in *Globalisierung*

Informationen zu Unicode stehen in der aktuellen Ausgabe von *The Unicode Standard* sowie auf der Website von The Unicode Consortium unter www.unicode.org zur Verfügung.

Speicherbedarf für Tabellen

Beim Entwerfen von Tabellen müssen Sie den Bedarf an Speicherplatz für die Daten, die in den Tabellen gespeichert werden sollen, mit einkalkulieren. Insbesondere müssen Sie auf Spalten mit größeren Datentypen wie LOB- oder XML-Daten achten.

LOB-Daten (LOB = Large Object, großes Objekt)

Daten großer Objekte (LOB-Daten) werden in zwei separaten Tabellenobjekten gespeichert, die eine andere Struktur als der Speicherbereich für andere Datentypen aufweisen. Bei der Abschätzung des für LOB-Daten erforderlichen Speicherbereichs müssen Sie die beiden Tabellenobjekte berücksichtigen, die zum Speichern von Daten dieser Datentypen verwendet werden:

- *LOB-Datenobjekte:* Die Daten werden in Bereichen von 64 MB gespeichert, die sich aus Segmenten zusammensetzen, deren Größen sich aus dem Produkt der Zweierpotenzen und 1024 Byte errechnen. (Diese Segmente können also aus 1024 Byte, 2048 Byte, 4096 Byte usw. bis 64 MB bestehen.)

  Zur Verringerung des für LOB-Daten erforderlichen Plattenspeicherplatzes können Sie den Parameter COMPACT in der Klausel für die LOB-Optionen der Anweisungen CREATE TABLE und ALTER TABLE angeben. Die Option COMPACT minimiert die Größe des für die LOB-Daten erforderlichen Speicherplatzes dadurch, dass die LOB-Daten in kleinere Segmente aufgeteilt werden können. Dieser Prozess beinhaltet keine Datenkomprimierung, sondern verwendet lediglich den kleinstmöglichen Speicherbereich bis zur nächsten 1-KB-Grenze. Die Angabe der Option COMPACT kann zu einer geringeren Leistung führen, wenn Daten an LOB-Werte angehängt werden.

  Der Umfang des freien Speicherbereichs innerhalb der LOB-Datenobjekte wird vom Umfang der Aktualisierungs- und Löschaktivitäten sowie von der Größe der eingefügten LOB-Werte beeinflusst.
- *LOB-Zuordnungsobjekte:* Die Informationen zur Zuordnung und zu freien Speicherbereichen werden in Zuordnungsseiten gespeichert, die von den eigentlichen Daten getrennt sind. Die Anzahl dieser Seiten ist vom Umfang der Daten

(einschließlich des nicht genutzten Speicherbereichs) abhängig, die für die LOB-Daten zugeordnet wurden. Der zusätzliche Speicherbedarf wird wie folgt berechnet:

Tabelle 22. Zusätzlicher Speicherbedarf bei Zuordnungsseiten auf der Basis der Seitengröße

Seitengröße	Zuordnungsseiten
4 KB	Eine Seite für jeweils 4 MB plus eine Seite für jeweils 1 GB
8 KB	Eine Seite für jeweils 8 MB plus eine Seite für jeweils 2 GB
16 KB	Eine Seite für jeweils 16 MB plus eine Seite für jeweils 4 GB
32 KB	Eine Seite für jeweils 32 MB plus eine Seite für jeweils 8 GB

Wenn Zeichendaten kleiner als die Seitengröße sind und sie zusammen mit dem Rest der Daten in den Datensatz passen, sollten anstelle der Datentypen BLOB, CLOB oder DBCLOB die Datentypen CHAR, GRAPHIC, VARCHAR oder VARGRAPHIC verwendet werden.

Anmerkung: Einige LOB-Daten können in der Basistabellenspalte gespeichert werden, wenn die Option INLINE LENGTH der Anweisungen CREATE TABLE und ALTER TABLE angegeben wird.

Langfelddaten (LF-Daten)

Langfelddaten (LF) werden in einem separaten Tabellenobjekt gespeichert, das eine andere Struktur als der Speicherbereich für andere Datentypen aufweist. Die Daten werden in Bereichen von 32 KB gespeichert, die sich aus Segmenten zusammensetzen, deren Größen sich aus dem Produkt der Zweierpotenzen und 512 Byte errechnen. (Diese Segmente können also aus 512 Byte, 1024 Byte, 2048 Byte usw. bis 32.768 Byte bestehen.)

Langfelddatentypen (LONG VARCHAR oder LONG VARGRAPHIC) werden auf eine Weise gespeichert, die eine problemlose Neuverwendung von freigegebenem Speicherbereich ermöglicht. Die Informationen zur Zuordnung und zu freien Speicherbereichen werden in Zuordnungsseiten von jeweils 4 KB gespeichert, die gelegentlich im Objekt erscheinen.

Der Bereich des freien Speicherplatzes in den Objekten ist von der Größe der Langfelddaten sowie davon abhängig, ob diese Größe bei jedem Vorkommen der Daten relativ konstant ist. Bei Dateneinträgen von mehr als 255 Byte kann dieser nicht genutzte Speicherplatz bis zu 50 Prozent der Größe der Langfelddaten ausmachen.

Wenn Zeichendaten kleiner als die Seitengröße sind und sie zusammen mit dem Rest der Daten in den Datensatz passen, sollten anstelle der Datentypen LONG VARCHAR oder LONG VARGRAPHIC die Datentypen CHAR, GRAPHIC, VARCHAR oder VARGRAPHIC verwendet werden.

Systemkatalogtabellen

Systemkatalogtabellen werden erstellt, wenn eine Datenbank erstellt wird. Der Umfang dieser Systemtabellen nimmt in dem Maße zu, wie der Datenbank Datenbankobjekte und Zugriffsrechte hinzugefügt werden. Zu Beginn belegen diese Tabellen einen Plattenspeicherplatz von ungefähr 3,5 MB.

Die Größe des Speicherbereichs, der für die Katalogtabellen zugeordnet wird, hängt von der Art des Tabellenbereichs und der Größe des durch EXTENTSIZE definierten Bereichs des Tabellenbereichs mit den Katalogtabellen ab. Wenn zum Bei-

spiel ein DMS-Tabellenbereich mit einem EXTENTSIZE-Wert von 32 verwendet wird, wird dem Katalogtabellenbereich anfangs ein Speicherbereich von 20 MB zugeordnet. Hinweis: Für Datenbanken mit mehreren Partitionen befinden sich die Katalogtabellen nur in der Datenbankpartition, von der aus der Befehl **CREATE DATABASE** abgesetzt wurde. Plattenspeicherplatz für die Katalogtabellen ist nur für diese Datenbankpartition erforderlich.

Temporäre Tabellen

Einige Anweisungen benötigen für die Verarbeitung temporäre Tabellen (z. B. eine Arbeitsdatei für Sortieroperationen, die nicht im Speicher vorgenommen werden können). Diese temporären Tabellen benötigen Plattenspeicherplatz. Die Größe des erforderlichen Speichers hängt von der Größe, der Anzahl und der Art der Abfragen sowie von der Größe der Ergebnistabellen ab.

Da Ihre Arbeitsumgebung einzigartig ist, lässt sich der Platzbedarf für temporäre Tabellen nur sehr schwer schätzen. Zum Beispiel kann es aufgrund der längeren Lebensdauer verschiedener temporärer Systemdateien so scheinen, als ob mehr Speicherplatz für Tabellenbereiche für temporäre Systemtabellen zugeordnet wäre, als in Wirklichkeit genutzt wird. Dies könnte der Fall sein, wenn die Registrierdatenbankvariable **DB2_SMS_TRUNC_TMPTABLE_THRESH** verwendet wird.

Mit dem Datenbanksystemmonitor können und den APIs zum Abfragen des Tabellenbereichs können Sie die Größe des Arbeitsbereichs verfolgen, der während des normalen Betriebsablaufs genutzt wird.

Mit der Registrierdatenbankvariablen **DB2_OPT_MAX_TEMP_SIZE** können Sie die Größe des Tabellenbereichs für temporäre Tabellen begrenzen, die von Abfragen verwendet wird.

XML-Daten

XML-Dokumente, die Sie in Spalten vom Typ XML einfügen, können sich entweder im Standardspeicherobjekt oder direkt in der Basistabellenzeile befinden. Das Speichern in Basistabellenzeilen unterliegt Ihrer Steuerung und ist nur für kleine Dokumente möglich. Größere Dokumente werden immer im Standardspeicherobjekt gespeichert.

Tabellenseitengrößen

Zeilen von Tabellendaten werden in Blöcken verwaltet, die als Seiten bezeichnet werden. Seiten können vier verschiedene Größen haben: 4, 8, 16 und 32 Kilobyte. Tabellendatenseiten enthalten keine Daten für Spalten, die mit den Datentypen LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB, DCLOB oder XML definiert sind. (Eine Ausnahme bilden nur LOB-Daten oder XML-Dokumente, die in einer Spalte unter Verwendung der Option INLINE LENGTH der Spalte 'inline' gespeichert werden.) Die Zeilen auf einer Tabellendatenseite enthalten jedoch einen Deskriptor dieser Spalten.

Anmerkung: Einige LOB- und XML-Daten können in der Basistabellenspalte gespeichert werden, wenn die Option INLINE LENGTH der Anweisungen CREATE TABLE und ALTER TABLE verwendet wird.

Sie können Pufferpools und Tabellenbereiche erstellen, die Seitengrößen von 4, 8, 16 oder 32 KB besitzen. Alle Tabellen, die innerhalb eines Tabellenbereichs einer be-

stimmten Größe erstellt wurden, besitzen eine übereinstimmende Seitengröße. Die maximale Größe für ein einzelnes Tabellen- oder Indexobjekt beträgt 64 TB bei einer Seitengröße von 32 KB.

Sie können maximal 1012 Spalten haben, wenn Sie eine Seitengröße von 8, 16 oder 32 KB haben. Bei einer Seitengröße von 4 KB sind maximal 500 Spalten möglich. Die Maximalanzahl von Zeilen, die pro Seite möglich sind, beträgt unabhängig von der Seitengröße 255.

Die maximalen Zeilenlängen sind je nach verwendeter Seitengröße unterschiedlich:

- Bei einer Seitengröße von 4 KB beträgt die maximale Zeilenlänge 4.005 Byte.
- Bei einer Seitengröße von 8 KB beträgt die maximale Zeilenlänge 8.101 Byte.
- Bei einer Seitengröße von 16 KB beträgt die maximale Zeilenlänge 16.293 Byte.
- Bei einer Seitengröße von 32 KB beträgt die maximale Zeilenlänge 32.677 Byte.

Zur Ermittlung der geeigneten Seitengröße für einen Tabellenbereich müssen Sie folgende Faktoren berücksichtigen:

- Für OLTP-Anwendungen, die wahlfreie Lese- und Schreiboperationen durchführen, ist meist eine geringere Seitengröße vorzuziehen, weil dadurch weniger Pufferspeicher durch unerwünschte Zeilen belegt wird.
- Für DSS-Anwendungen, die jeweils auf eine große Anzahl aufeinander folgender Zeilen gleichzeitig zugreifen, sind in der Regel größere Seiten vorteilhafter, weil dadurch weniger E/A-Anforderungen erforderlich sind, um eine bestimmte Anzahl Zeilen zu lesen. Es gibt jedoch eine Ausnahme von dieser Regel. Wenn die von Ihnen verwendete Zeilengröße kleiner als `PAGESIZE / maximale Anzahl Zeilen` ist, bleibt auf jeder Seite Speicherbereich ungenutzt. In diesem Fall ist eine geringere Seitengröße möglicherweise geeigneter.

Durch größere Seiten können Sie möglicherweise die Zahl der Indexstufen reduzieren. Größere Seiten unterstützen längere Zeilen. Bei Verwendung von 4-KB-Seiten sind Tabellen auf 500 Spalten begrenzt. Größere Seitengrößen (8, 16 und 32 KB) unterstützen bis zu 1012 Spalten. Die Maximalgröße des Tabellenbereichs ist proportional zur Seitengröße des Tabellenbereichs.

Speicherbedarf für Benutzertabellendaten

Standardmäßig werden Tabellendaten auf der Basis der Seitengröße des Tabellenbereichs gespeichert, in dem sich die Tabelle befindet. Jede Seite enthält (unabhängig von der Seitengröße) 68 Byte Systemaufwand für den Datenbankmanager. Eine Zeile kann sich *nicht* über mehrere Seiten erstrecken. Wenn Sie Seiten mit jeweils 4 KB verwenden, sind maximal 500 Spalten möglich.

Tabellendatenseiten enthalten *keine* Daten für Spalten, die mit den Datentypen LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB oder XML definiert sind. Die Zeilen einer Tabellendatenseite enthalten jedoch einen Deskriptor für diese Spalten.

Anmerkung: Einige LOB-Daten können in der Basistabellenspalte gespeichert werden, wenn die Option INLINE LENGTH der Anweisungen CREATE TABLE und ALTER TABLE angegeben wird.

Zeilen werden normalerweise an der ersten passenden Stelle in einer regulären Tabelle eingefügt. Die Datei wird nach dem ersten verfügbaren Speicherbereich durchsucht (unter Verwendung einer Liste der freien Speicherbereiche), der groß genug ist, um die neue Zeile aufzunehmen. Das Aktualisieren einer vorhandenen Zeile erfolgt an der ursprünglichen Stelle, es sei denn, der freie Bereich der 4-KB-

Seite reicht nicht aus, um die aktualisierte Zeile aufzunehmen. In diesem Fall wird an der Position, an der sich die Originalzeile befand, ein Datensatz erstellt, der auf die neue Speicherposition der aktualisierten Zeile in der Tabellendatei verweist.

Bei Verwendung der Anweisung ALTER TABLE mit der Option `APPEND ON` werden Daten immer angehängt, und es werden keine Informationen zum freien Speicherbereich der Datenseiten aufgezeichnet.

Wenn die Tabelle über einen definierten Clusterindex verfügt, versucht der Datenbankmanager, die Daten entsprechend der Schlüsselreihenfolge dieses Clusterindex physisch (zu Clustern) zusammenzufassen. Wenn eine Zeile in die Tabelle eingefügt wird, sucht der Datenbankmanager zuerst den entsprechenden Schlüsselwert im Clusterindex. Wird der Schlüsselwert gefunden, versucht der Datenbankmanager, den Datensatz auf der durch den Schlüssel angegebenen Datenseite einzufügen. Wird der Schlüsselwert nicht gefunden, wird der nächst höhere Schlüsselwert verwendet, sodass der Datensatz auf der Seite eingefügt wird, die Datensätze mit dem nächst höheren Schlüsselwert enthält. Wenn auf der Zielseite nicht genügend freier Speicherbereich zur Verfügung steht, wird die Liste der freien Speicherbereiche zur Suche nach benachbarten Seiten mit freiem Speicherbereich verwendet. Mit der Zeit wird der Speicherbereich auf den Datenseiten vollständig belegt, und Datensätze werden immer weiter von der Zielseite entfernt in der Tabelle gespeichert. Schließlich werden die Tabellendaten als Daten ohne Clustering betrachtet. In diesem Fall kann die Clusterreihenfolge durch eine Tabellenreorganisation wiederhergestellt werden.

Wenn es sich bei der Tabelle um eine MDC-Tabelle (mit mehrdimensionalem Clustering) handelt, garantiert der Datenbankmanager, dass Datensätze immer entlang mindestens einer definierten Dimension oder entsprechend den Clusterindizes physisch in Clustern angeordnet werden. Wenn eine MDC-Tabelle mit bestimmten Dimensionen definiert wird, wird ein Blockindex für jede der Dimensionen sowie ein zusammengesetzter Blockindex erstellt, der Zellen (eindeutige Kombinationen aus Dimensionswerten) Blöcken zuordnet. Dieser zusammengesetzte Blockindex dient zur Bestimmung, zu welcher Zelle ein bestimmter Datensatz gehört und welche Blöcke oder EXTENTSIZE großen Speicherbereiche in der Tabelle Datensätze enthalten, die zu dieser Zelle gehören. Beim Einfügen von Datensätzen durchsucht der Datenbankmanager daher den zusammengesetzten Blockindex nach der Liste von Blöcken, die Datensätze mit den gleichen Dimensionswerten enthalten, und beschränkt die Suche nach freiem Speicherplatz auf diese Blöcke. Wenn die Zelle noch nicht vorhanden ist oder wenn in den vorhandenen Blöcken der Zelle kein ausreichender Speicherbereich zur Verfügung steht, wird der Zelle ein weiterer Block zugeordnet und der Datensatz in diesen eingefügt. Eine Liste der freien Speicherbereiche wird weiterhin innerhalb von Blöcken verwendet, um rasch verfügbaren Speicherbereich in den Blöcken ausfindig zu machen.

Die Anzahl der 4-KB-Seiten für jede Benutzertabelle in der Datenbank kann nach folgender Berechnung geschätzt werden:

```
ABRUNDEN(4028/(durchschnittszeilengröße + 10)) = datensätze_pro_seite
```

Das Ergebnis wird in folgende Berechnung eingefügt:

```
(zahl_der_datensätze/datensätze_pro_seite) * 1,1 = anzahl_der_seiten
```

Dabei ist die durchschnittliche Zeilenlänge die Summe der durchschnittlichen Spaltengrößen, und der Faktor "1,1" dient zur Kalkulation des Systemaufwands.

Anmerkung: Diese Formel liefert nur einen Schätzwert. Die Genauigkeit des Schätzwerts nimmt ab, wenn die Datensatzlänge durch Fragmentierung und Überlaufsätze variiert.

Sie können auch Pufferpools oder Tabellenbereiche mit einer Seitengröße von 8, 16 oder 32 KB erstellen. Alle Tabellen, die innerhalb eines Tabellenbereichs einer bestimmten Größe erstellt werden, besitzen eine übereinstimmende Seitengröße. Die maximale Größe für ein einzelnes Tabellen- oder Indexobjekt beträgt 64 TB bei einer Seitengröße von 32 KB. Wenn Sie Seiten mit einer Größe von 8, 16 oder 32 KB verwenden, sind maximal 1012 Spalten möglich. Für eine 4-KB-Seite beträgt die maximale Spaltenanzahl 500. Die maximalen Zeilenlängen variieren ebenfalls je nach Seitengröße:

- Bei einer Seitengröße von 4 KB beträgt die maximale Zeilenlänge 4.005 Byte.
- Bei einer Seitengröße von 8 KB beträgt die maximale Zeilenlänge 8.101 Byte.
- Bei einer Seitengröße von 16 KB beträgt die maximale Zeilenlänge 16.293 Byte.
- Bei einer Seitengröße von 32 KB beträgt die maximale Zeilenlänge 32.677 Byte.

Größere Seiten ermöglichen eine geringere Anzahl von Indexstufen in einem Index. Wenn Sie mit OLTP-Anwendungen arbeiten, die wahlfreie Lese- und Schreibzugriffe auf Zeilen durchführen, ist eine geringere Seitengröße empfehlenswert, weil weniger Pufferspeicher durch unerwünschte Zeilen belegt wird. Wenn Sie mit DSS-Anwendungen (Decision Support System) arbeiten, die jeweils auf eine große Anzahl aufeinander folgender Zeilen zugreifen, sind größere Seiten empfehlenswert, weil dann weniger E/A-Anforderungen erforderlich sind, um eine bestimmte Anzahl Zeilen zu lesen.

Ein Backup-Image kann nicht in einer anderen Seitengröße wiederherstellt werden.

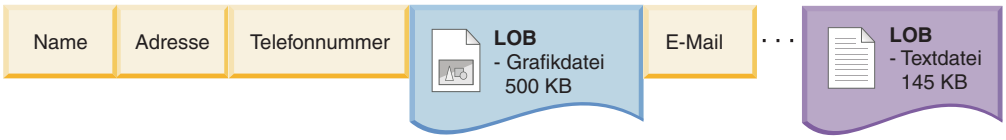
Sie können keine IXF-Datendateien importieren, die mehr als 755 Spalten beinhalten.

Deklarierte oder erstellte temporäre Tabellen können nur in einem eigenen Tabellenbereichstyp für temporäre Benutzertabellen deklariert bzw. erstellt werden. Es gibt keinen Standardtabellenbereich für temporäre Benutzertabellen. Die temporären Tabellen werden implizit gelöscht, wenn eine Anwendung die Verbindung zur Datenbank trennt. Schätzungen über den Platzbedarf solcher Tabellen sollten dies berücksichtigen.

Speichern von LOBs inline in Tabellenzeilen

Große Objekte (LOBs, Large Objects) werden im Allgemeinen an einer Position gespeichert, die von der Tabellenzeile, die auf sie verweist, getrennt ist. Sie haben jedoch die Möglichkeit, ein großes Objekt mit einer Länge von bis zu 32.673 Byte inline in einer Basistabellenzeile zu speichern, um den Zugriff auf dieses Objekt zu vereinfachen.

Es kann unpraktisch (und abhängig von den Daten auch unmöglich) sein, große Datenobjekte in Basistabellenzeilen unterzubringen. Abb. 35 auf Seite 338 zeigt ein Beispiel für den Versuch, LOB-Daten in einer Zeile zu speichern, und warum dies ein Problem sein kann. In diesem Beispiel ist die Zeile mit zwei LOB-Spalten von 500 bzw. 145 KB Länge definiert. Die maximale Zeilengröße für eine DB2-Tabelle beträgt jedoch 32 KB. Daher könnte eine solche Zeilendefinition tatsächlich nie implementiert werden.



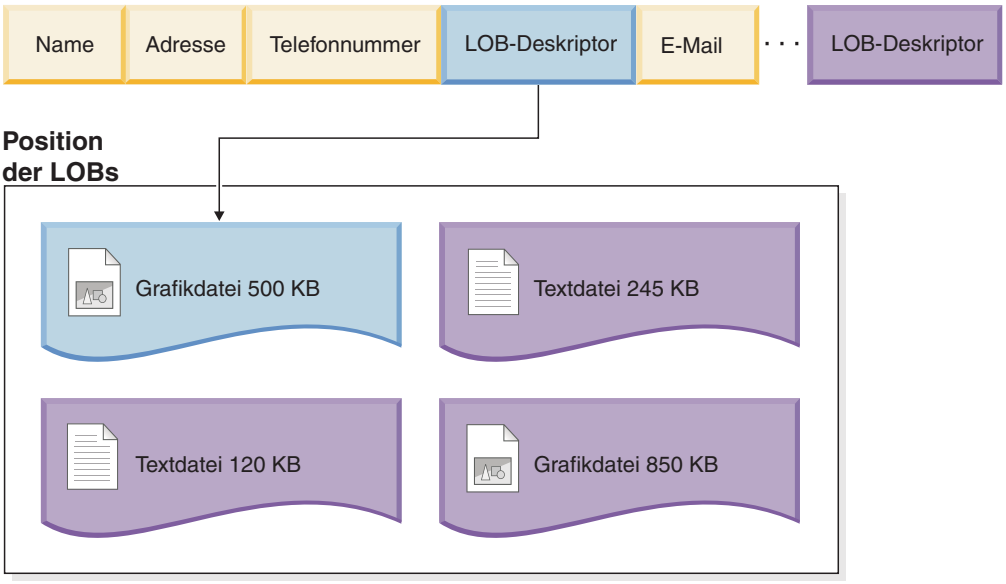
Legende

LOB = Große Objekte

Abbildung 35. Problem beim Speichern von LOB-Daten in Basistabellenzeilen

Zur Verringerung der Schwierigkeiten, die mit der Arbeit mit LOB-Daten verbunden sind, werden diese Daten anders als andere Datentypen behandelt. Abb. 36 zeigt, dass nur ein LOB-Deskriptor in die Basistabellenzeile eingefügt wird, und nicht das LOB-Objekt selbst. Alle LOB-Objekte werden ihrerseits an einer separaten LOB-Position gespeichert, die vom Datenbankmanager gesteuert wird. Bei dieser Anordnung benötigt das Versetzen von Zeilen zwischen dem Pufferpool und dem Plattenspeicher weniger Zeit für Zeilen mit LOB-Deskriptoren, als wenn die Zeilen die vollständigen LOB-Objekte enthielten.

Die Bearbeitung der LOB-Daten wird in diesem Fall jedoch schwieriger, da die tatsächlichen LOB-Daten an einer von den Basistabellenzeilen getrennten Position gespeichert werden.



Legende

LOB = Große Objekte

Abbildung 36. LOB-Deskriptoren in einer Basistabellenzeile verweisen auf die LOBs an der separaten LOB-Position

Zur Vereinfachung der Bearbeitung von kleineren LOBs haben Sie die Möglichkeit, LOB-Daten, die unter einem Größenschwellenwert liegen, den Sie angeben, inline

in die Basistabellenzeilen einzufügen. Solche LOB-Datentypen können anschließend als Teil der Basistabellenzeile bearbeitet werden, sodass sich Operationen, wie zum Beispiel die Versetzung in und aus dem Pufferpool, einfacher gestalten. Darüber hinaus kommen inline gespeicherte LOB-Daten für die Zeilenkomprimierung infrage, sofern die Zeilenkomprimierung aktiviert wurde.

Durch die Option INLINE LENGTH der Anweisungen CREATE TABLE und ALTER TABLE können LOB-Daten, die kleiner als die von Ihnen angegebene Längenbeschränkung sind, in die Basistabellenzeile eingefügt werden. Standardmäßig werden LOB-Daten, die kleiner als die maximale Größe für LOB-Deskriptoren für die Spalte sind, immer in die Basistabellenzeile eingefügt.

Bei inline gespeicherten LOB-Daten können Basistabellenzeilen wie in Abb. 37 gezeigt aussehen.

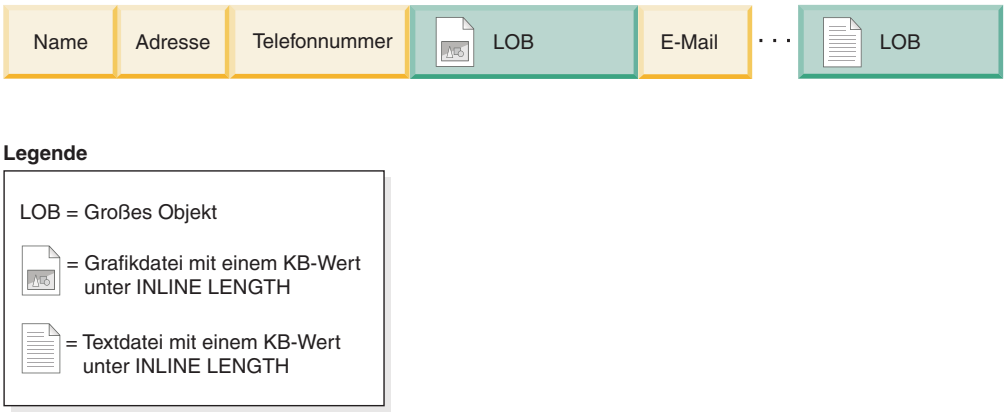


Abbildung 37. Kleine, in Basistabellenzeilen eingefügte LOB-Daten

Wenn Sie sich überlegen, welcher Schwellenwert für das Inlinespeichern von LOBs auszuwählen ist, müssen Sie die aktuelle Seitengröße für Ihre Datenbank berücksichtigen und klären, ob die Zeilengröße durch inline gespeicherte LOB-Daten die aktuelle Seitengröße überschreiten würde. Die maximale Größe für eine Zeile in einer Tabelle beträgt 32.677 Byte. Jedoch ist jedes inline gespeicherte LOB-Objekt mit 4 Byte an zusätzlichem Speicherbedarf verbunden. Daher verringert jedes LOB-Objekt, das Sie inline speichern, den verfügbaren Speicherplatz in der Zeile um 4 Byte. Infolgedessen beträgt die maximale Größe für ein Inline-LOB-Objekt 32.673 Byte.

Anmerkung: Auf dieselbe Weise, wie LOB-Daten inline gespeichert werden können, ist es auch möglich, XML-Daten inline zu speichern.

Tabellenkomprimierung

Sie können den für Ihre Tabellen erforderlichen Plattenspeicherplatz verringern, indem Sie die DB2-Funktionen zur Tabellenkomprimierung nutzen. Die Komprimierung ermöglicht Plattenspeichereinsparungen, indem weniger Datenbankseiten zum Speichern von Daten verwendet werden.

Da Sie mehr Zeilen pro Seite speichern können, müssen außerdem für den Zugriff auf das gleiche Datenvolumen weniger Seiten gelesen werden. Daher werden für Abfragen für eine komprimierte Tabelle weniger E/A-Operationen für den Zugriff auf das gleiche Datenvolumen benötigt. Da auf einer Pufferpoolseite mehr Datenzeilen vorhanden sind, steigt die Wahrscheinlichkeit, dass sich benötigte Zeilen im

Pufferpool befinden. Daher kann eine Komprimierung aufgrund verbesserter Pufferpooltrefferquoten eine Leistungssteigerung bewirken. Ähnlich lassen sich mithilfe von Komprimierung Backup- und Restoreoperationen beschleunigen, da für die Sicherung und Wiederherstellung des gleichen Datenvolumens weniger Seiten übertragen werden müssen.

Sie können die Komprimierung sowohl bei neuen als auch bei vorhandenen Tabellen verwenden. Temporäre Tabellen werden auch automatisch komprimiert, wenn der Datenbankmanager der Ansicht ist, dies sei vorteilhaft.

Für Tabellen stehen zwei Haupttypen der Datenkomprimierung zur Verfügung:

- Zeilenkomprimierung (verfügbar mit einer Lizenz für das DB2 Storage Optimization Feature).
- Wertkomprimierung

Sie können für eine bestimmte Tabelle die Zeilenkomprimierung und die Wertkomprimierung zusammen oder einzeln verwenden. Sie können jedoch für eine bestimmte Tabelle nur einen einzigen Zeilenkomprimierungstyp verwenden.

Zeilenkomprimierung

Die Zeilenkomprimierung verwendet einen wörterverzeichnisbasierten Komprimierungsalgorithmus, um wiederholt auftretende Zeichenfolgen innerhalb von Datenzeilen durch kürzere Symbole zu ersetzen.

Es stehen zwei Typen der Zeilenkomprimierung zur Auswahl:

- „Klassische" Zeilenkomprimierung.
- Adaptive Komprimierung

Zeilenkomprimierung ist mit einer Lizenz für DB2 Storage Optimization Feature verfügbar. Je nach der Ihnen zur Verfügung stehenden DB2-Produktversion ist diese Komponente möglicherweise enthalten oder sie kann eine Option sein, die Sie separat bestellen.

Klassische Zeilenkomprimierung:

Die klassische Zeilenkomprimierung, die manchmal auch als *statische Komprimierung* bezeichnet wird, komprimiert Datenzeilen, indem sie Muster von Werten, die sich über Zeilen hinweg wiederholen, durch kürzere Symbolzeichenfolgen ersetzt.

Die Vorteile der klassischen Zeilenkomprimierung ähneln denen der adaptiven Komprimierung und bestehen darin, dass Sie Daten in weniger Speicherplatz speichern können, sodass sich erhebliche Einsparungen bei den Speicherkosten ergeben können. Im Gegensatz zur adaptiven Komprimierung verwendet die klassische Zeilenkomprimierung jedoch nur ein Wörterverzeichnis auf Tabellenebene, um Muster, die global wiederholt auftreten, zu speichern. Sie verwendet nicht die Wörterverzeichnisse auf Seitenebene, die zum dynamischen Komprimieren von Daten verwendet werden.

Funktionsweise der klassischen Zeilenkomprimierung

Die klassische Zeilenkomprimierung verwendet ein Komprimierungswörterverzeichnis auf Tabellenebene, um Daten zeilenweise zu komprimieren. Das Wörterverzeichnis wird verwendet, um sich wiederholende Bytemuster aus Tabellenzeilen wesentlich kleineren Symbolen zuzuordnen. Diese Symbole werden dann anstelle der längeren Bytemuster in die Tabellenzeilen eingefügt. Das Komprimierungswör-

terverzeichnis wird zusammen mit den Tabellendatenzeilen in den Datenobjektabschnitten der Tabelle gespeichert.

Welche Daten werden komprimiert?

Daten, die in Basistabellenzeilen gespeichert sind, und Protokollsätze kommen für die klassische Zeilenkomprimierung in Frage. Darüber hinaus ist eine Komprimierung von Daten in XML-Speicherobjekten möglich. LOB-Daten, die Sie integriert in einer Tabellenzeile ('inline') speichern, können komprimiert werden. Allerdings werden Speicherobjekte für lange Datenobjekte nicht komprimiert.

Einschränkung: Sie können Daten in XML-Spalten, die mit DB2 Version 9.5 oder DB2 Version 9.1 erstellt wurden, nicht komprimieren. Sie können jedoch Inline-XML-Spalten komprimieren, die Sie mit DB2 Version 9.7 oder höher einer Tabelle hinzufügen, sofern die Tabelle in einem früheren Produktrelease ohne XML-Spalten erstellt wurde. Wenn eine Tabelle, die Sie in einem früheren Release erstellt haben, bereits mindestens eine XML-Spalte hat und Sie eine komprimierte XML-Spalte mit DB2 Version 9.7 oder höher hinzufügen wollen, müssen Sie die Tabelle mithilfe der gespeicherten Prozedur ADMIN_MOVE_TABLE migrieren, bevor Sie die Komprimierung verwenden können.

Aktivieren oder Inaktivieren der klassischen Zeilenkomprimierung

Zur Verwendung der klassischen Zeilenkomprimierung müssen Sie eine Lizenz für das DB2 Storage Optimization Feature haben. Sie komprimieren Tabellendaten, indem Sie das Attribut COMPRESS der Tabelle auf den Wert YES STATIC setzen. Sie können dieses Attribut definieren, wenn Sie die Tabelle erstellen, indem Sie für die Anweisung CREATE TABLE die Option COMPRESS YES STATIC angeben. Darüber hinaus können Sie eine vorhandene Tabelle zur Verwendung der Komprimierung ändern, indem Sie dieselbe Option für die Anweisung ALTER TABLE verwenden. Wenn die Komprimierung aktiviert ist, kann die klassische Zeilenkomprimierung von Operationen verwendet werden, die der Tabelle Daten hinzufügen, wie zum Beispiel **INSERT**, **LOAD INSERT** oder **IMPORT INSERT**. Darüber hinaus wird für die Tabelle die Indexkomprimierung aktiviert. Indizes werden als komprimierte Indizes erstellt, sofern Sie nichts anderes angeben und es sich um Typen von Indizes handelt, die komprimiert werden können.

Wichtig: Wenn Sie die klassische Zeilenkomprimierung für eine Tabelle aktivieren, gilt dies für die gesamte Tabelle, selbst wenn eine Tabelle mehrere Tabellenpartitionen umfasst.

Zur Inaktivierung der Komprimierung für eine Tabelle verwenden Sie die Anweisung ALTER TABLE mit der Option COMPRESS NO. Zeilen, die Sie nachfolgend hinzufügen, werden nicht komprimiert. Zum Extrahieren der gesamten Tabelle müssen Sie eine Tabellenreorganisation mit dem Befehl **REORG TABLE** ausführen.

Wenn Sie über eine Lizenz für DB2 Storage Optimization Feature verfügen, wird die Komprimierung für temporäre Dateien automatisch aktiviert. Sie können die Komprimierung für temporäre Tabellen nicht aktivieren oder inaktivieren.

Auswirkungen von Aktualisierungsaktivitäten auf Protokolle und komprimierte Tabellen

Abhängig von den Aktualisierungsaktivitäten und Spalten, die innerhalb einer Datenzeile aktualisiert werden, kann die Protokollbelegung ansteigen. Informationen zum Minimieren der Auswirkungen von Aktualisierungsaktivitäten auf Protokolle

finden Sie im Abschnitt „„Ordnen von Spalten zur Minimierung der Aktualisierungsprotokollierung" auf Seite 330".

Wenn eine Zeile wächst, ist es möglich, dass die neue Version der Zeile nicht auf die aktuelle Datenseite passt. Stattdessen wird das neue Image der Zeile auf einer Überlaufseite gespeichert. Damit möglichst wenige Datensätze mit Verweisen auf Überlaufzeilen erstellt werden, erhöhen Sie nach einer Reorganisation mithilfe der Option PCTFREE für die Anweisung ALTER TABLE den Prozentsatz, der auf jeder Seite als freier Speicher zu behalten ist. Wenn die Option PCTFREE beispielsweise auf 5 % gesetzt war, bevor Sie die Komprimierung aktiviert haben, könnten Sie sie auf 10 % setzen, wenn Sie die Komprimierung aktivieren. Die Erhöhung des auf jeder Seite als freier Speicher zu behaltenden Prozentsatzes ist besonders bei Daten wichtig, die häufig aktualisiert werden.

Klassische Zeilenkomprimierung für temporäre Tabellen

Die Komprimierung für temporäre Tabellen wird automatisch durch das DB2 Storage Optimization Feature aktiviert. Bei der Ausführung von Abfragen zieht das DB2-Optimierungsprogramm die Speichereinsparungen und die Auswirkungen auf die Abfrageleistung in Betracht, die eine Komprimierung temporärer Tabellen bewirken würde, um zu ermitteln, ob eine Verwendung der Komprimierung sinnvoll wäre. Wenn sie sinnvoll ist, wird die Komprimierung automatisch verwendet. Die Mindestgröße, die eine Tabelle erreichen muss, bevor die Komprimierung verwendet wird, ist für temporäre Tabellen größer als reguläre Tabellen.

Mithilfe der EXPLAIN-Funktion oder des Tools **db2pd** können Sie ermitteln, ob das Optimierungsprogramm die Komprimierung für temporäre Tabellen verwendet hat.

Verfügbarmachen des durch Komprimierung freigegebenen Speicherplatzes

Sie können den Speicherplatz, der durch die Komprimierung von Daten freigegeben wurde, für das System verfügbar machen. Weitere Informationen finden Sie in „Konsolidierbarer Speicherbereich" auf Seite 195.

Adaptive Komprimierung:

Adaptive Komprimierung verbessert die Komprimierungsraten, die bei alleiniger Verwendung der klassischen Zeilenkomprimierung erreicht werden können. Adaptive Komprimierung beinhaltet die klassische Zeilenkomprimierung, sie arbeitet jedoch auch Seite für Seite, um Daten noch stärker zu komprimieren. Von den verschiedenen Datenkomprimierungsverfahren im DB2-Produkt bietet die adaptive Komprimierung das beträchtlichste Potenzial für Speichereinsparungen.

Funktionsweise der adaptiven Komprimierung

Bei der adaptiven Komprimierung werden eigentlich zwei Komprimierungsverfahren verwendet. Das erste Verfahren komprimiert Daten mithilfe desselben Komprimierungswörterverzeichnisses auf Tabellenebene, das bei der klassischen Zeilenkomprimierung verwendet wird, auf der Basis von Wiederholungen innerhalb einer Stichprobe von Daten aus der gesamten Tabelle. Bei dem zweiten Verfahren werden die Daten mithilfe eines Komprimierungsalgorithmus, der auf Wörterverzeichnissen auf Seitenebene aufbaut, auf der Basis von Datenwiederholungen innerhalb der einzelnen Datenseiten komprimiert. Die Wörterverzeichnisse ordnen sich wiederholende Bytemuster wesentlich kleineren Symbolen zu. Diese Symbole werden dann anstelle der längeren Bytemuster in die Tabelle eingefügt. Das Kom-

primierungswörterverzeichnis auf Tabellenebene wird in dem Tabellenobjekt gespeichert, für das es erstellt wurde, und wird zur Datenkomprimierung in der gesamten Tabelle verwendet. Das Komprimierungswörterverzeichnis auf Seitenebene wird mit den Daten auf der Datenseite gespeichert und wird ausschließlich zur Komprimierung der Daten auf dieser Seite verwendet. Weitere Informationen zur Rolle dieser Wörterverzeichnisse bei der Datenkomprimierung finden Sie im Abschnitt „Komprimierungswörterverzeichnisse" auf Seite 350.

Anmerkung: Sie können die Komprimierung einer Tabelle mit klassischer Zeilenkomprimierung nur angeben, wenn Sie ein Komprimierungswörterverzeichnis auf Tabellenebene verwenden. Sie können jedoch nicht angeben, dass Tabellen ausschließlich mit Komprimierungswörterverzeichnissen auf Seitenebene komprimiert werden sollen. Bei der adaptiven Komprimierung werden sowohl Komprimierungswörterverzeichnisse auf Tabellenebene als auch auf Seitenebene verwendet.

Für Komprimierung infrage kommende Daten

Daten, die *in* Datenzeilen gespeichert werden, einschließlich von Inline-LOB- oder Inline-XML-Daten, können sowohl mit der adaptiven als auch der klassischen Zeilenkomprimierung komprimiert werden. XML-Speicherobjekte können mithilfe der statischen Komprimierung komprimiert werden. Speicherobjekte für lange Datenobjekte, die außerhalb von Tabellenzeilen gespeichert werden, werden jedoch nicht komprimiert. Obwohl Protokollsätze selbst nicht komprimiert werden, wird darüber hinaus der Umfang von Protokolldaten aus Einfüge-, Aktualisierungs- oder Löschoperationen aufgrund der Zeilen verringert, die komprimiert werden.

Einschränkung: Sie können Daten in XML-Spalten, die mit DB2 Version 9.5 oder DB2 Version 9.1 erstellt wurden, nicht komprimieren. Sie können jedoch Inline-XML-Spalten komprimieren, die Sie mit DB2 Version 9.7 oder höher einer Tabelle hinzufügen, sofern die Tabelle in einem früheren Produktrelease ohne XML-Spalten erstellt wurde. Wenn eine Tabelle, die Sie in einem früheren Release erstellt haben, bereits mindestens eine XML-Spalte hat und Sie eine komprimierte XML-Spalte mit DB2 Version 9.7 oder höher hinzufügen wollen, müssen Sie die Tabelle mithilfe der gespeicherten Prozedur ADMIN_MOVE_TABLE migrieren, bevor Sie die Komprimierung verwenden können.

Aktivieren und Inaktivieren der adaptiven Komprimierung

Zur Verwendung der adaptiven Komprimierung müssen Sie eine Lizenz für das DB2 Storage Optimization Feature haben. Sie komprimieren Tabellendaten, indem Sie das Attribut COMPRESS der Tabelle auf den Wert YES setzen. Sie können dieses Attribut definieren, wenn Sie die Tabelle erstellen, indem Sie für die Anweisung CREATE TABLE die Option COMPRESS YES angeben. Darüber hinaus können Sie eine vorhandene Tabelle zur Verwendung der Komprimierung ändern, indem Sie dieselbe Option für die Anweisung ALTER TABLE verwenden. Wenn die Komprimierung aktiviert ist, kann die adaptive Komprimierung von Operationen verwendet werden, die der Tabelle Daten hinzufügen, wie zum Beispiel **INSERT**, **LOAD INSERT** oder **IMPORT INSERT**. Darüber hinaus wird für die Tabelle die Indexkomprimierung aktiviert. Indizes werden als komprimierte Indizes erstellt, sofern Sie nichts anderes angeben und es sich um Typen von Indizes handelt, die komprimiert werden können.

Wichtig: Wenn Sie die adaptive Komprimierung für eine Tabelle aktivieren, aktivieren Sie sie für die gesamte Tabelle, selbst wenn die Tabelle mehrere Tabellenpartitionen umfasst.

Zur Inaktivierung der Komprimierung für eine Tabelle verwenden Sie die Anweisung ALTER TABLE mit der Option COMPRESS NO. Zeilen, die Sie später hinzufügen, werden nicht komprimiert. Vorhandene Zeilen bleiben komprimiert. Zum Extrahieren der gesamten Tabelle nach dem Inaktivieren der Komprimierung müssen Sie mit dem Befehl **REORG TABLE** eine Tabellenreorganisation ausführen.

Wenn Sie die Lizenz für DB2 Storage Optimization Feature anwenden, wird die Komprimierung für temporäre Tabellen automatisch aktiviert, wenn der Datenbankmanager dies für sinnvoll erachtet. Sie können die Komprimierung für temporäre Tabellen nicht aktivieren oder inaktivieren.

Auswirkungen von Aktualisierungsaktivitäten auf Protokolle und komprimierte Tabellen

Abhängig von den Aktualisierungsaktivitäten und der Position der Aktualisierungen in einer Datenzeile kann die Protokollbelegung ansteigen. Informationen zu den Auswirkungen der Reihenfolge von Spalten in einer Tabelle auf die Aktualisierungsprotokollierung finden Sie im Abschnitt „„Ordnen von Spalten zur Minimierung der Aktualisierungsprotokollierung” auf Seite 330”.

Wenn eine Zeile nach dem Hinzufügen neuer Daten größer wird, passt die neue Version der Zeile möglicherweise nicht auf die aktuelle Datenseite. Stattdessen wird das neue Image der Zeile auf einer Überlaufseite gespeichert. Damit möglichst wenige Datensätze mit Verweisen auf Überlaufzeilen erstellt werden, erhöhen Sie nach einer Reorganisation mithilfe der Option PCTFREE für die Anweisung ALTER TABLE den Prozentsatz, der auf jeder Seite als freier Speicher zu behalten ist. Wenn die Option PCTFREE beispielsweise auf 5 % gesetzt war, bevor Sie die Komprimierung aktiviert haben, könnten Sie sie auf 10 % setzen, wenn Sie die Komprimierung aktivieren. Die Erhöhung des auf jeder Seite als freier Speicher zu behaltenden Prozentsatzes ist besonders bei Daten wichtig, die häufig aktualisiert werden.

Komprimierung für temporäre Tabellen

Die Komprimierung für temporäre Tabellen wird automatisch durch das DB2 Storage Optimization Feature aktiviert. Für temporäre Tabellen wird nur die klassische Zeilenkomprimierung verwendet.

Temporäre Systemtabellen
: Bei der Ausführung von Abfragen zieht das DB2-Optimierungsprogramm die Speichereinsparungen und die Auswirkung auf die Abfrageleistung in Betracht, die eine Komprimierung vom System erstellter temporärer Tabellen bewirken würde, um zu ermitteln, ob eine Verwendung der Komprimierung sinnvoll ist. Falls es sinnvoll ist, wird die klassische Zeilenkomprimierung automatisch verwendet. Die Mindestgröße, die eine Tabelle erreichen muss, bevor die Komprimierung verwendet wird, ist für temporäre Tabellen größer als reguläre Tabellen.

Benutzererstellte temporäre Tabellen
: Erstellte globale temporäre Tabellen (CGTTs) und deklarierte globale temporäre Tabellen (DGTTs) werden stets mithilfe der klassischen Zeilenkomprimierung komprimiert.

Mithilfe der EXPLAIN-Funktion oder des Tools **db2pd** können Sie ermitteln, ob das Optimierungsprogramm die Komprimierung für temporäre Systemtabellen verwendet hat.

Verfügbarmachen des durch Komprimierung freigegebenen Speicherplatzes

Sie können den Speicherplatz, der durch die Komprimierung von Daten freigegeben wurde, für das System verfügbar machen. Weitere Informationen finden Sie in „Konsolidierbarer Speicherbereich" auf Seite 195.

Schätzen von Speichereinsparungen durch adaptive Komprimierung oder klassische Zeilenkomprimierung

Sie können eine Schätzung der Speichereinsparungen, die durch die adaptive Komprimierung oder die klassische Zeilenkomprimierung für eine Tabelle erzielt werden können, mithilfe der Tabellenfunktion ADMIN_GET_TAB_COMPRESS_INFO anzeigen.

Vorbereitende Schritte

Die geschätzten Einsparungen, die sich durch die adaptive Komprimierung oder die klassische Zeilenkomprimierung realisieren lassen, hängen von den Statistiken ab, die durch Ausführung des Befehls **RUNSTATS** generiert werden. Zur Ermittlung einer möglichst genauen Schätzung der erzielbaren Einsparungen führen Sie den Befehl **RUNSTATS** aus, bevor Sie die folgenden Schritte ausführen.

Vorgehensweise

Gehen Sie wie folgt vor, um die Speichereinsparungen durch die adaptive Komprimierung oder die klassische Zeilenkomprimierung mithilfe der Tabellenfunktion ADMIN_GET_TAB_COMPRESS_INFO zu schätzen:

1. Formulieren Sie eine SELECT-Anweisung, die die Tabellenfunktion ADMIN_GET_TAB_COMPRESS_INFO verwendet. Für eine Tabelle mit dem Namen SAMPLE1.T1 könnten Sie zum Beispiel Folgendes eingeben:

   ```
   SELECT * FROM TABLE(SYSPROC.ADMIN_GET_TAB_COMPRESS_INFO('SAMPLE1', 'T1'))
   ```

2. Führen Sie die Anweisung SELECT aus. Die Ausführung der in Schritt 1 gezeigten Anweisung könnte zum Beispiel einen Bericht wie den folgenden liefern:

   ```
   TABSCHEMA TABNAME       DBPARTITIONNUM DATAPARTITIONID OBJECT_TYPE ROWCOMPMODE ...
   --------- ------------- -------------- --------------- ----------- ---------- ...
   SAMPLE1   T1            0              0               DATA        A          ...

     1 Satz/Sätze ausgewählt.

   PCTPAGESSAVED_CURRENT AVGROWSIZE_CURRENT PCTPAGESSAVED_STATIC AVGROWSIZE_STATIC ...
   --------------------- ------------------ -------------------- ----------------- ...
                      96                 24                   81               148 ...

   PCTPAGESSAVED_ADAPTIVE AVGROWSIZE_ADAPTIVE
   ---------------------- -------------------
                       93                  44
   ```

Erstellen einer Tabelle mit Komprimierung

Wenn Sie mit der Anweisung CREATE TABLE eine neue Tabelle erstellen, haben Sie die Option, die in den Tabellenzeilen enthaltenen Daten zu komprimieren.

Vorbereitende Schritte

Sie müssen entscheiden, welcher Komprimierungstyp verwendet werden soll: die adaptive Komprimierung, die klassische Zeilenkomprimierung, die Wertkomprimierung oder eine Kombination aus Wertkomprimierung und einer der beiden Zeilenkomprimierungstypen. Bei der adaptiven Komprimierung und der klassischen Zeilenkomprimierung wird in fast allen Fällen Speicher eingespart, da bei diesen

Typen versucht wird, Datenmuster, die sich über mehrere Spalten erstrecken, durch kürzere Symbolzeichenfolgen zu ersetzen. Die Wertkomprimierung kann zu Einsparungen führen, wenn viele Zeilen mit Spalten vorhanden sind, die denselben Wert enthalten, beispielsweise einen Städte- oder Landesnamen, oder wenn Spalten vorhanden sind, die den Standardwert für den Datentyp der Spalte enthalten.

Vorgehensweise

Setzen Sie eine Anweisung CREATE TABLE ab, um eine Tabelle zu erstellen, die Komprimierung verwendet.

- Wenn Sie die adaptive Komprimierung verwenden wollen, geben Sie die Klausel COMPRESS YES ADAPTIVE an.
- Wenn Sie die klassische Zeilenkomprimierung verwenden wollen, geben Sie die Klausel COMPRESS YES STATIC an.
- Wenn Sie die Wertkomprimierung verwenden wollen, geben Sie die Klausel VALUE COMPRESSION an. Wenn Daten komprimiert werden sollen, die Systemstandardwerte für Spalten darstellen, geben Sie außerdem die Klausel COMPRESS SYSTEM DEFAULT an.

Ergebnisse

Nach der Erstellung der Tabelle werden alle Daten komprimiert, die Sie ab diesem Zeitpunkt zu der Tabelle hinzufügen. Auch alle der Tabelle zugeordneten Indizes werden komprimiert, sofern Sie nicht mithilfe der Klausel COMPRESS NO der Anweisung CREATE INDEX oder ALTER INDEX etwas anderes festlegen.

Beispiele

Beispiel 1: Mit der folgenden Anweisung wird eine Tabelle für Kundeninformation mit aktivierter adaptiver Komprimierung erstellt. In diesem Beispiel werden zur Komprimierung der Tabelle sowohl Komprimierungswörterverzeichnisse auf Tabellenebene als auch auf Seitenebene verwendet.

```
CREATE TABLE CUSTOMER
     (CUSTOMERNUM      INTEGER,
      CUSTOMERNAME     VARCHAR(80),
      ADDRESS          VARCHAR(200),
      CITY             VARCHAR(50),
      COUNTRY          VARCHAR(50),
      CODE             VARCHAR(15),
      CUSTOMERNUMDIM   INTEGER)
      COMPRESS YES ADAPTIVE;
```

Beispiel 2: Mit der folgenden Anweisung wird eine Tabelle für Kundeninformation mit klassischer Zeilenkomprimierung erstellt. In diesem Beispiel wird zur Komprimierung der Tabelle nur ein Komprimierungswörterverzeichnis auf Tabellenebene verwendet.

```
CREATE TABLE CUSTOMER
     (CUSTOMERNUM      INTEGER,
      CUSTOMERNAME     VARCHAR(80),
      ADDRESS          VARCHAR(200),
      CITY             VARCHAR(50),
      COUNTRY          VARCHAR(50),
      CODE             VARCHAR(15),
      CUSTOMERNUMDIM   INTEGER)
      COMPRESS YES STATIC;
```

Beispiel 3: Mit der folgenden Anweisung wird eine Tabelle für Mitarbeitergehälter erstellt. Die Spalte SALARY hat den Standardwert 0 und für die Spalte werden die Zeilenkomprimierung und die Komprimierung der Systemstandardwerte angegeben.

```
CREATE TABLE EMPLOYEE_SALARY
     (DEPTNO    CHAR(3)     NOT NULL,
      DEPTNAME VARCHAR(36)  NOT NULL,
      EMPNO    CHAR(6)      NOT NULL,
      SALARY   DECIMAL(9,2) NOT NULL WITH DEFAULT COMPRESS SYSTEM DEFAULT)
      COMPRESS YES ADAPTIVE;
```

Beachten Sie, dass in dieser Anweisung die Klausel VALUE COMPRESSION weggelassen wurde. Durch diese Anweisung wird eine Tabelle mit dem Namen EMPLOYEE_SALARY erstellt, jedoch auch eine Warnung zurückgegeben:

```
SQL20140W  COMPRESS column attribute ignored because VALUE COMPRESSION is
deactivated for the table.  SQLSTATE=01648
```

In diesem Fall wird die Klausel COMPRESS SYSTEM DEFAULT nicht auf die Spalte SALARY angewendet.

Beispiel 4: Mit der folgenden Anweisung wird eine Tabelle für Mitarbeitergehälter erstellt. Die Spalte SALARY hat den Standardwert 0 und für die Spalte werden die Zeilenkomprimierung und die Komprimierung der Systemstandardwerte aktiviert.

```
CREATE TABLE EMPLOYEE_SALARY
     (DEPTNO    CHAR(3)     NOT NULL,
      DEPTNAME VARCHAR(36)  NOT NULL,
      EMPNO    CHAR(6)      NOT NULL,
      SALARY   DECIMAL(9,2) NOT NULL WITH DEFAULT COMPRESS SYSTEM DEFAULT)
      VALUE COMPRESSION COMPRESS YES ADAPTIVE;
```

In diesem Beispiel ist die Klausel VALUE COMPRESSION in der Anweisung enthalten. Daher wird der Standardwert des Felds SALARY komprimiert.

Aktivieren der Komprimierung in einer vorhandenen Tabelle

Sie können eine vorhandene Tabelle mithilfe des Befehls ALTER TABLE ändern, um die speichersparenden Vorteile der Komprimierung zu nutzen.

Vorbereitende Schritte

Sie müssen entscheiden, welcher Komprimierungstyp verwendet werden soll: die adaptive Komprimierung, die klassische Zeilenkomprimierung, die Wertkomprimierung oder eine Kombination aus Wertkomprimierung und einer der beiden Zeilenkomprimierungstypen. Bei der adaptiven Komprimierung und der klassischen Zeilenkomprimierung wird in fast allen Fällen Speicher eingespart, da bei diesen Typen versucht wird, Datenmuster, die sich über mehrere Spalten erstrecken, durch kürzere Symbolzeichenfolgen zu ersetzen. Die Wertkomprimierung kann zu Einsparungen führen, wenn viele Zeilen mit Spalten vorhanden sind, die denselben Wert enthalten, beispielsweise einen Städte- oder Landesnamen, oder wenn Spalten vorhanden sind, die den Standardwert für den Datentyp der Spalte enthalten.

Vorgehensweise

Gehen Sie wie folgt vor, um in einer vorhandenen Tabelle die Komprimierung zu aktivieren:

1. Führen Sie die Anweisung ALTER TABLE aus.
   - Wenn Sie die adaptive Komprimierung verwenden wollen, geben Sie die Klausel COMPRESS YES ADAPTIVE an.

- Wenn Sie die klassische Zeilenkomprimierung verwenden wollen, geben Sie die Klausel COMPRESS YES STATIC an.
- Wenn Sie die Wertkomprimierung verwenden wollen, geben Sie die Klausel ACTIVATE VALUE COMPRESSION für jede Spalte an, die einen zu komprimierenden Wert enthält. Wenn Daten in Spalten komprimiert werden sollen, die Systemstandardwerte enthalten, geben Sie außerdem die Klausel COMPRESS SYSTEM DEFAULT an.

Alle Zeilen, die Sie danach anhängen, einfügen, laden oder aktualisieren, verwenden das neue komprimierte Format.

2. Optional: Wenn die Komprimierung sofort auf alle vorhandenen Zeilen einer Tabelle angewendet werden soll, müssen Sie mithilfe des Befehls **REORG TABLE** eine Tabellenreorganisation ausführen. Wenn Sie an dieser Stelle die Komprimierung nicht auf alle Zeilen anwenden, werden die nicht komprimierten Zeilen erst dann in dem neuen komprimierten Format gespeichert, wenn Sie sie das nächste Mal aktualisieren oder das nächste Mal den Befehl REORG TABLE ausführen.

Beispiele

Beispiel 1: Mit der folgenden Anweisung wird die adaptive Komprimierung auf eine vorhandene Tabelle namens CUSTOMER angewendet:

```
ALTER TABLE CUSTOMER      COMPRESS YES ADAPTIVE
```

Beispiel 2: Mit der folgenden Anweisung wird die klassische Komprimierung auf eine vorhandene Tabelle namens CUSTOMER angewendet:

```
ALTER TABLE CUSTOMER      COMPRESS YES STATIC
```

Beispiel 3: Mit den folgenden Anweisungen wird die Komprimierung für Zeilen, Werte und Systemstandardwerte auf die Spalte SALARY einer vorhandenen Tabelle namens EMPLOYEE_SALARY angewendet. Anschließend wird die Tabelle reorganisiert.

```
ALTER TABLE EMPLOYEE_SALARY
ALTER SALARY COMPRESS SYSTEM DEFAULT
COMPRESS YES ACTIVATE VALUE COMPRESSION;

REORG TABLE EMPLOYEE_SALARY
```

Ändern oder Inaktivieren der Komprimierung einer komprimierten Tabelle

Sie können die Art der Komprimierung einer Tabelle ändern oder die Komprimierung für eine Tabelle mit aktivierter adaptiver Komprimierung, klassischer Zeilenkomprimierung oder Wertkomprimierung komplett inaktivieren. Verwenden Sie dazu eine oder mehrere der für die Steuerung der Komprimierung verfügbaren Klauseln der Anweisung ALTER TABLE.

Informationen zu diesem Vorgang

Wenn Sie die adaptive oder die klassische Zeilenkomprimierung inaktivieren, ist die Indexkomprimierung davon nicht betroffen. Wenn Sie die Komprimierung eines Index inaktivieren wollen, müssen Sie die Anweisung ALTER INDEX verwenden.

Vorgehensweise

Gehen Sie wie folgt vor, um die Komprimierung für eine Tabelle zu inaktivieren oder den Typ der Zeilenkomprimierung zu ändern:

1. Führen Sie eine Anweisung ALTER TABLE aus.
   - Wenn Sie die adaptive Komprimierung oder die klassische Zeilenkomprimierung inaktivieren wollen, geben Sie die Klausel COMPRESS NO an.
   - Wenn Sie den Typ der Zeilenkomprimierung ändern wollen, geben Sie mit der Klausel COMPRESS YES ADAPTIVE oder COMPRESS YES STATIC den gewünschten Komprimierungstyp an. Wenn Sie beispielsweise eine Tabelle haben, in der gerade die klassische Zeilenkomprimierung verwendet wird, und Sie die adaptive Komprimierung festlegen wollen, führen Sie die Anweisung ALTER TABLE mit der Klausel COMPRESS YES ADAPTIVE aus.
   - Wenn Sie die Wertkomprimierung inaktivieren wollen, geben Sie die Klausel DEACTIVATE VALUE COMPRESSION an.
   - Wenn Sie die Komprimierung von Systemstandardwerten inaktivieren wollen, geben Sie die Option COMPRESS OFF in der Klausel ALTER *spaltenname* an.
2. Führen Sie mit dem Befehl **REORG TABLE** eine Offlinetabellenreorganisation aus.

Ergebnisse

- Wenn Sie die Zeilenkomprimierung mit der Klausel COMPRESS NO inaktiviert haben, werden alle Zeilendaten dekomprimiert.
- Wenn Sie den Typ der Zeilenkomprimierung geändert haben, wird die gesamte Tabelle mit dem Typ der Zeilenkomprimierung komprimiert, den Sie in der Anweisung ALTER TABLE angegeben haben. (Siehe Beispiel 2.)
- Die Inaktivierung der Wertkomprimierung hat folgende Auswirkungen:
  - Wenn eine Tabelle Spalten mit aktivierter Option COMPRESS SYSTEM DEFAULT enthielt, wird die Komprimierung dieser Spalten nicht weiter fortgesetzt.
  - Nicht komprimierte Spalten können dazu führen, dass die Zeilengröße das zulässige Maximum für die aktuelle Seitengröße des aktuellen Tabellenbereichs überschreitet. In diesem Fall wird die Fehlernachricht SQL0670N zurückgegeben.

Beispiele

Beispiel 1: Inaktivieren der Zeilenkomprimierung: Die folgenden Anweisungen inaktivieren die adaptive oder die klassische Zeilenkomprimierung in einer Tabelle namens CUSTOMER und reorganisieren die Tabelle anschließend so, dass die zuvor komprimierten Daten dekomprimiert werden:

```
ALTER TABLE CUSTOMER COMPRESS NO
REORG TABLE CUSTOMER
```

Beispiel 2: Wechseln von statischer zu adaptiver Komprimierung: Dabei wird angenommen, dass für die Tabelle SALES derzeit die klassische Zeilenkomprimierung verwendet wird. Mit den folgenden Anweisungen wird der verwendete Komprimierungstyp in adaptive Komprimierung geändert:

```
ALTER TABLE SALES COMPRESS ADAPTIVE YES
REORG TABLE SALES
```

Komprimierungswörterverzeichnisse

Der Datenbankmanager erstellt ein Komprimierungswörterverzeichnis auf Tabellenebene für jede Tabelle, die Sie für adaptive oder klassische Zeilenkomprimierung aktivieren. Bei Tabellen, die Sie für adaptive Komprimierung aktivieren, erstellt der Datenbankmanager außerdem Komprimierungswörterverzeichnisse auf Seitenebene.

Beide Wörterverzeichnistypen werden verwendet, um sich wiederholende Bytemuster aus Tabellenzeilen wesentlich kleineren Symbolen zuzuordnen. Diese Symbole werden dann anstelle der längeren Bytemuster in die Tabellenzeilen eingefügt.

Komprimierungswörterverzeichnisse auf Tabellenebene

Zum Erstellen von Wörterverzeichnissen auf Tabellenebene wird die Tabelle nach wiederholt auftretenden Mustern gescannt. Es werden vollständige Zeilen, nicht nur bestimmte Felder oder Teile von Zeilen, auf sich wiederholende Einträge bzw. Muster geprüft. Nach der Erfassung der sich wiederholenden Einträge erstellt der Datenbankmanager ein Komprimierungswörterverzeichnis, wobei den Einträgen kurze numerische Schlüssel zugeordnet werden. Im Allgemeinen bieten Textzeichenfolgen größere Chancen zur Komprimierung als numerische Daten. Bei der Komprimierung numerischer Daten wird eine Zahl durch eine andere ersetzt. Je nach der Größe der zu ersetzenden Zahlen sind die Speichereinsparungen unter Umständen nicht so umfangreich wie bei der Komprimierung von Text.

Die erste Erstellung eines Wörterverzeichnisses auf Seitenebene erfolgt mithilfe einer Stichprobe aus der Tabelle. Das Wörterverzeichnis wird nicht mehr aktualisiert, sofern Sie keinen expliziten Rebuild des Wörterverzeichnisses mithilfe der klassischen Offlinetabellenreorganisation anstoßen. Selbst bei einem Rebuild des Wörterverzeichnisses enthält dieses nur eine Stichprobe der Daten der gesamten Tabelle.

Hinweis: Das Wörterverzeichnis auf Tabellenebene ist statisch. Es wird nach seiner Erstellung nicht geändert, außer Sie führen einen manuellen Rebuild aus. Auch im Falle eines Rebuilds werden aufgrund der Verfahren zur Stichprobenentnahme, die bei der Erstellung des Wörterverzeichnisses verwendet werden, möglicherweise keine Zeichenfolgen eingeschlossen, die auf einer einzelnen Seite wiederholt auftreten.

Das Komprimierungswörterverzeichnis auf Tabellenebene wird in verdeckten Zeilen in dem Objekt gespeichert, auf das sich diese beziehen, und wird für den schnellen Zugriff im Hauptspeicher zwischengespeichert. Dieses Wörterverzeichnis belegt nicht viel Speicherplatz. Selbst für sehr große Tabellen benötigt das Komprimierungswörterverzeichnis normalerweise nur ca. 100 KB an Speicherplatz.

Komprimierungswörterverzeichnisse auf Seitenebene

Bei der adaptiven Komprimierung werden neben Wörterverzeichnissen auf Tabellenebene auch Wörterverzeichnisse auf Seitenebene verwendet. Im Gegensatz zu Wörterverzeichnissen auf Tabellenebene werden Wörterverzeichnisse auf Seitenebene automatisch erstellt oder neu erstellt, wenn Seiten vom Datenbankmanager gefüllt werden. Ebenso wie Komprimierungswörterverzeichnisse auf Tabellenebene werden Wörterverzeichnisse auf Seitenebene auch in verdeckten Zeilen in der Tabelle gespeichert.

Erstellen von Komprimierungswörterverzeichnissen auf Tabellenebene

Komprimierungswörterverzeichnisse auf Tabellenebene für Tabellen, die Sie für adaptive oder klassische Zeilenkomprimierung aktivieren, können automatisch oder manuell erstellt werden. Zu den Tabellen, die Sie für adaptive Komprimierung aktivieren, gehören Datenwörterverzeichnisse auf Seitenebene, die stets automatisch erstellt werden.

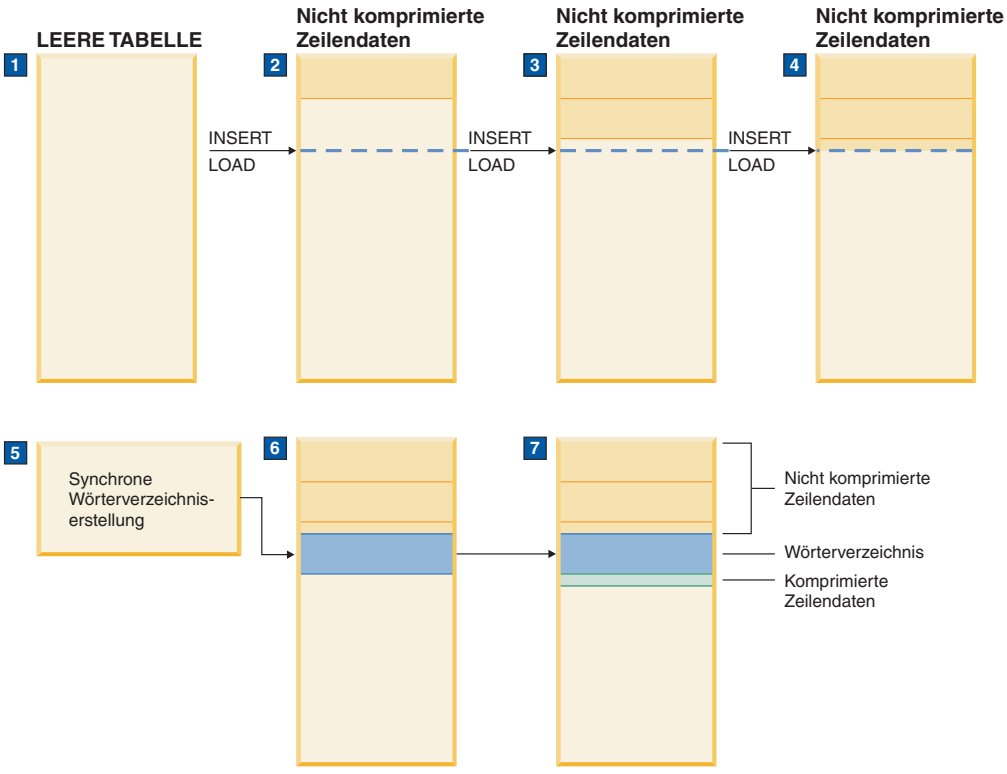
Automatische Erstellung von Komprimierungswörterverzeichnissen

Ab DB2 Version 9.5 wird ein Komprimierungswörterverzeichnis automatisch erstellt, wenn jede der folgenden Bedingungen zutrifft:

- Sie setzen das Attribut COMPRESS für die Tabelle, indem Sie die Anweisung CREATE TABLE oder ALTER TABLE mit der Klausel COMPRESS YES ADAPTIVE oder COMPRESS YES STATIC verwenden.
- Es ist noch kein Komprimierungswörterverzeichnis auf Tabellenebene für die Tabelle vorhanden.
- Die Tabelle enthält genügend Daten zur Erstellung eines Wörterverzeichnisses mit wiederholt vorkommenden Daten.

Daten, die Sie nach der Erstellung des Wörterverzeichnisses in die Tabelle versetzen, werden mithilfe des Wörterverzeichnisses komprimiert, wenn die Komprimierung aktiviert bleibt.

Das folgende Diagramm zeigt den Prozess der automatischen Erstellung des Komprimierungswörterverzeichnisses:



In dem Diagramm wird die folgende Ereignisreihenfolge dargestellt:

1. Es wird noch kein Komprimierungswörterverzeichnis erstellt, da die Tabelle leer ist.

2. Daten werden durch INSERT- oder LOAD-Operationen in die Tabelle eingefügt und bleiben unkomprimiert.
3. Wenn weitere Daten in die Tabelle eingefügt oder geladen werden, bleiben diese unkomprimiert.
4. Nach Erreichen eines Schwellenwerts wird die Wörterverzeichniserstellung automatisch ausgelöst, wenn das Attribut COMPRESS auf den Wert YES ADAPTIVE oder YES STATIC gesetzt ist.
5. Das Wörterverzeichnis wird erstellt.
6. Das Wörterverzeichnis wird an die Tabelle angehängt.
7. Von diesem Punkt an wird die Komprimierung auf Tabellenebene aktiviert und die zu einem späteren Zeitpunkt eingefügten oder hinzugefügten Zeilen werden durch das Komprimierungswörterverzeichnis auf Tabellenebene komprimiert.

Wichtig: Die Zeilen, die vor der Erstellung des Wörterverzeichnisses in einer Tabelle vorhanden waren, bleiben unkomprimiert, sofern Sie sie nicht ändern oder das Wörterverzeichnis manuell neu erstellen.

Wenn eine Tabelle mit DB2 Version 9.7 oder höher erstellt wird und mindestens eine Spalte vom Typ XML enthält, wird ein zweites Komprimierungswörterverzeichnis erstellt. Mithilfe dieses Wörterverzeichnisses werden die XML-Daten komprimiert, die im Standard-XML-Speicherobjekt gespeichert sind, das der Tabelle zugeordnet ist. Die Erstellung von Komprimierungswörterverzeichnissen für XML-Daten erfolgt automatisch, wenn alle folgenden Bedingungen erfüllt sind:

- Sie haben das Attribut COMPRESS für die Tabelle auf YES ADAPTIVE oder YES STATIC gesetzt.
- In dem entsprechenden XML-Speicherobjekt ist kein Komprimierungswörterverzeichnis vorhanden.
- In dem XML-Speicherobjekt sind genügend Daten vorhanden.

Einschränkung: Sie können Daten in XML-Spalten, die mit DB2 Version 9.5 oder DB2 Version 9.1 erstellt wurden, nicht komprimieren. Sie können jedoch Inline-XML-Spalten komprimieren, die Sie mit DB2 Version 9.7 oder höher einer Tabelle hinzufügen, sofern die Tabelle in einem früheren Produktrelease ohne XML-Spalten erstellt wurde. Wenn eine Tabelle, die Sie in einem früheren Release erstellt haben, bereits mindestens eine XML-Spalte hat und Sie eine komprimierte XML-Spalte mit DB2 Version 9.7 oder höher hinzufügen wollen, müssen Sie die Tabelle mithilfe der gespeicherten Prozedur ADMIN_MOVE_TABLE migrieren, bevor Sie die Komprimierung verwenden können.

Der Mechanismus zum Erstellen von Komprimierungswörterverzeichnissen auf Tabellenebene für temporäre Tabellen ähnelt dem Mechanismus, der für permanente Tabellen verwendet wird. Allerdings bestimmt der Datenbankmanager automatisch, ob die klassische Zeilenkomprimierung für temporäre Tabellen genutzt wird, auf der Basis von Faktoren wie der Abfragekomplexität und der Größe der Ergebnismenge.

Manuelle Erstellung von Komprimierungswörterverzeichnissen

Obwohl Komprimierungswörterverzeichnisse automatisch erstellt werden, wenn Tabellen mit aktivierter Komprimierung eine ausreichende Größe erreichen, können Sie mithilfe des Befehls **REORG TABLE** und des Parameters **RESETDICTIONARY** die Erstellung eines Komprimierungswörterverzeichnisses auf Tabellenebene auch erzwingen, wenn keines vorhanden ist. Dieser Befehl erzwingt die Erstellung eines

Komprimierungswörterverzeichnisses, wenn sich mindestens eine Zeile mit Daten in der Tabelle befindet. Die Tabellenreorganisation ist eine offline ausgeführte Operation. Ein Vorteil der automatischen Wörterverzeichniserstellung besteht darin, dass die Tabelle online verfügbar bleibt, während das Wörterverzeichnis erstellt wird.

Anstatt die Erstellung eines neuen Wörterverzeichnisses mit dem Befehl **REORG TABLE** zu erzwingen, können Sie auch den Befehl **INSPECT** mit dem Parameter **ROWCOMPESTIMATE** verwenden. Dieser Befehl erstellt ein Komprimierungswörterverzeichnis, wenn die Tabelle nicht bereits über eines verfügt. Der Vorteil dieser Methode gegenüber der Ausführung einer Tabellenreorganisation ist, dass die Tabelle online verfügbar bleibt. Zeilen, die Sie zu einem späteren Zeitpunkt hinzufügen, unterliegen der Komprimierung, während Zeilen, die vor der Ausführung des Befehls **INSPECT** vorhanden waren, unkomprimiert bleiben, bis Sie eine Tabellenreorganisation ausführen. Wenn die Komprimierung jedoch aktiviert ist, findet die automatische Wörterverzeichniserstellung in der Regel kurz nach der Aktivierung der Komprimierung statt, sogar noch bevor Sie überhaupt die Möglichkeit haben, den Befehl **INSPECT** zu verwenden.

Zurücksetzen von Komprimierungswörterverzeichnissen

Wenn ein Komprimierungswörterverzeichnis automatisch oder manuell erstellt wird, ist es statisch, das heißt, nach seiner Erstellung wird es nicht mehr geändert. Wenn Zeilen hinzugefügt oder aktualisiert werden, werden sie auf der Grundlage der Daten komprimiert, die im Komprimierungswörterverzeichnis vorhanden sind. Dieses Verhalten ist für viele Fälle geeignet. Betrachten Sie zum Beispiel eine Tabelle in einer Datenbank, die zur Verwaltung von Kundenkonten für einen städtischen Wasserversorger verwendet wird. Eine solche Tabelle könnte Spalten wie ADRESSE, ORT, BUNDESLAND, TELEFON, PLZ und KONTOTYP enthalten. Wenn ein Komprimierungswörterverzeichnis mit Daten aus einer solchen Tabelle erstellt wird, sind mit hoher Wahrscheinlichkeit auch bei einer nicht allzu großen Tabelle ausreichend Informationen, die sich wiederholen, für die klassische Zeilenkomprimierung vorhanden, um bedeutsame Platzeinsparungen zu erzielen. Möglicherweise sind viele der Daten von Kunde zu Kunde gleich, beispielsweise die Werte für ORT, PLZ oder BUNDESLAND oder Teile des Werts in der Spalte ADRESSE oder TELEFON.

Andere Tabellen wiederum ändern sich im Verlauf der Zeit möglicherweise beträchtlich. Betrachten Sie zum Beispiel eine Tabelle, die wie folgt für Einzelhandelsumsatzdaten verwendet wird:

- Eine Originaltabelle wird zum Sammeln der Daten auf monatlicher Basis verwendet.
- Jeden Monat wird ein neuer Datensatz in die Tabelle geladen.

In diesem Fall würde ein Komprimierungswörterverzeichnis, das zum Beispiel im April erstellt wurde, möglicherweise nicht die sich wiederholenden Daten aus Umsätzen in späteren Teilen des Jahres berücksichtigen. In Situationen, in denen sich Daten in einer Tabelle im Verlauf der Zeit erheblich ändern, bietet es sich an, die Komprimierungswörterverzeichnisse durch den Befehl **REORG TABLE** mit dem Parameter **RESETDICTIONARY** zurückzusetzen. Die Zurücksetzung des Komprimierungswörterverzeichnisses bietet den Vorteil, dass bei Erstellung des Wörterverzeichnisses die gesamte Tabelle berücksichtigt wird.

Auswirkungen der klassischen Tabellenreorganisation auf Komprimierungswörterverzeichnisse auf Tabellenebene

Wenn Sie eine Tabelle, für die die adaptive Komprimierung oder die klassische Zeilenkomprimierung aktiviert ist, mithilfe der klassischen Offlinetabellenreorganisation reorganisieren, können Sie das Komprimierungswörterverzeichnis auf Tabellenebene beibehalten oder den Datenbankmanager anweisen, ein neues zu erstellen.

In DB2 Version 9.5 und höher wird automatisch ein Komprimierungswörterverzeichnis für eine Tabelle erstellt, für die Sie die adaptive oder klassische Zeilenkomprimierung aktivieren, indem Sie die Anweisung CREATE TABLE oder ALTER TABLE mit der Unterklausel **COMPRESS YES** verwenden. Bei einer neuen Tabelle wartet der Datenbankmanager, bis die Tabelle auf eine Mindestgröße anwächst, bevor das Wörterverzeichnis erstellt wird. Für eine vorhandene Tabelle wird das Komprimierungswörterverzeichnis erstellt, wenn die Tabelle auf eine ausreichende Größe anwächst, sodass Musterwiederholungen erkennbar werden. Die Komprimierung wird nur auf Zeilen angewendet, die Sie nach der Aktivierung der Komprimierung einfügen oder aktualisieren.

Wenn Sie eine Tabelle mithilfe einer klassischen Tabellenreorganisation reorganisieren und ein Komprimierungswörterverzeichnis auf Tabellenebene vorhanden ist, wird der Parameter **KEEPDICTIONARY** des Befehls **REORG TABLE** implizit angewendet und dadurch das Wörterverzeichnis beibehalten. Bei der Ausführung der Reorganisation unterliegen alle Zeilen, die verarbeitet werden, der Komprimierung unter Verwendung dieses Wörterverzeichnisses. Wenn kein Komprimierungswörterverzeichnis vorhanden ist und die Tabelle eine ausreichende Größe hat, wird ein Komprimierungswörterverzeichnis erstellt und die Zeilen unterliegen der Komprimierung unter Verwendung dieses Wörterverzeichnisses.

Sie können die Erstellung eines neuen Komprimierungswörterverzeichnisses auf Tabellenebene erzwingen, indem Sie eine klassische Tabellenreorganisation mit dem Parameter **RESETDICTIONARY** im Befehl **REORG TABLE** ausführen. Wenn Sie den Parameter **RESETDICTIONARY** angeben, wird ein neues Komprimierungswörterverzeichnis anstelle des vorhandenen Wörterverzeichnisses erstellt, wenn die Tabelle mindestens eine Zeile enthält.

Anmerkung: Wörterverzeichnisse auf Tabellenebene können nur mithilfe der klassischen Tabellenreorganisation erneut erstellt werden. Wenn Sie versuchen, die Inplace-Tabellenreorganisation einer Tabelle, in der Zeilen komprimiert sind, mithilfe eines Komprimierungswörterverzeichnisses auf Seitenebene durchzuführen, schlägt der Befehl REORG mit dem Fehler SQL2219 fehl.

Mehrere Komprimierungswörterverzeichnisse für Replikationsquellentabellen

Sie können bei den Anweisungen CREATE TABLE und ALTER TABLE die Klausel DATA CAPTURE CHANGES mit der Option COMPRESS YES STATIC oder COMPRESS YES ADAPTIVE kombinieren und so die Zeilenkomprimierung für Quellentabellen für die Replikation aktivieren.

Wenn Sie die Komprimierung aktivieren und außerdem die Klausel DATA CAPTURE CHANGES als Teil des Befehls **REORG TABLE** oder **LOAD REPLACE** angeben, kann eine Quellentabelle zwei Komprimierungswörterverzeichnisse auf Tabellenebene haben: ein aktives *Komprimierungswörterverzeichnis auf Tabellenebene* und ein *früheres Komprimierungswörterverzeichnis*. Somit wird bei aktivierter Klausel DATA CAPTURE CHANGES das Komprimierungswörterverzeichnis auf Tabellenebene

nicht ersetzt, wenn Sie den Befehl **REORG TABLE** oder **LOAD REPLACE** ausführen. Stattdessen wird ein neues Wörterverzeichnis generiert und das frühere Wörterverzeichnis wird beibehalten.

Dank des früheren Komprimierungswörterverzeichnisses kann die API db2ReadLog den Zeileninhalt in Protokollsätzen extrahieren, die vor dem Rebuild des Wörterverzeichnisses durch Verwenden der Option **RESETDICTIONARY** mit dem Befehl **REORG TABLE** oder **LOAD** geschrieben wurden.

Anmerkung: Wenn Protokolllesefunktionen die Daten in Protokollsätzen in einem nicht komprimierten Format anstatt in einem unaufbereiteten komprimierten Format zurückgeben sollen, müssen Sie den Parameter **iFilterOption** der API db2ReadLog auf den Wert DB2READLOG_FILTER_ON setzen.

Wenn Sie die Option DATA CAPTURE NONE in der Anweisung CREATE TABLE angegeben haben, mit der die Tabelle erstellt wurde, und anschließend den Befehl **REORG TABLE** absetzen oder durch Absetzen des Befehls **LOAD REPLACE**, **IMPORT REPLACE** oder **TRUNCATE TABLE** Operationen zum Abschneiden einer Tabelle ausführen, wird das frühere Komprimierungswörterverzeichnis für die Tabelle entfernt.

Prüfen Sie die Spalte HISTORICAL_DICTIONARY in der Ergebnismenge der Tabellenfunktion ADMIN_GET_TAB_DICTIONARY_INFO, um festzustellen, ob ein früheres Wörterverzeichnis für die Tabelle vorhanden ist.

Wertkomprimierung

Die Wertkomprimierung optimiert die Speicherbelegung für die Darstellung von Daten sowie die Speicherstrukturen, die vom Datenbankmanagementsystem intern zum Speichern von Daten verwendet werden. Bei der Wertkomprimierung werden doppelte Einträge für einen Wert entfernt, sodass nur eine Kopie gespeichert wird. Die gespeicherte Kopie zeichnet die Positionen aller Verweise auf den gespeicherten Wert auf.

Bei der Erstellung einer Tabelle können Sie mit der optionalen Klausel VALUE COMPRESSION der Anweisung CREATE TABLE angeben, dass die Tabelle die Wertkomprimierung verwenden soll. In einer vorhandenen Tabelle können Sie die Wertkomprimierung mit der Klausel ACTIVATE VALUE COMPRESSION der Anweisung ALTER TABLE aktivieren. Zur Inaktivierung der Wertkomprimierung in einer Tabelle verwenden Sie die Klausel DEACTIVATE VALUE COMPRESSION der Anweisung ALTER TABLE.

Wenn die Klausel VALUE COMPRESSION verwendet wird, werden Nullwerte (NULL) und Daten der Länge 0, die Datentypen mit variabler Länge (VARCHAR, VARGRAPHICS, LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB und DBCLOB) zugeordnet wurden, nicht auf der Platte gespeichert.

Bei Verwendung der Klausel VALUE COMPRESSION kann auch die Option COMPRESS SYSTEM DEFAULT verwendet werden, um die Plattenspeichernutzung weiter zu verringern. Eingefügte oder aktualisierte Werte, die mit dem Systemstandardwert für den Datentyp ihrer Spalte übereinstimmen, beanspruchen nur minimalen Plattenspeicherplatz, da der Standardwert nicht auf der Platte gespeichert wird. Alle numerischen Datentypen, Zeichenfolgetypen mit fester Länge und Grafikzeichenfolgen mit fester Länge für Spalten unterstützen die Option COMPRESS SYSTEM DEFAULT. Das heißt, dass Nullen und Leerzeichen komprimiert werden können.

Bei Verwendung der Wertkomprimierung kann die Bytezahl einer komprimierten Spalte in einer Zeile größer als die unkomprimierte Version derselben Spalte sein. Wenn sich die Größe einer Zeile dem zulässigen Maximalwert für die verwendete Seitengröße nähert, müssen Sie sicherstellen, dass die Summe der Bytezahlen für komprimierte und unkomprimierte Spalten nicht die zulässige Zeilenlänge der Tabelle im Tabellenbereich überschreitet. In einem Tabellenbereich mit einer Seitengröße von 4 KB beträgt die zulässige Zeilenlänge zum Beispiel 4005 Byte. Falls die zulässige Zeilenlänge überschritten wird, wird Fehlernachricht SQL0670N zurückgegeben. Die Formel, die zur Ermittlung der Bytezahlen für komprimierte und unkomprimierte Spalten verwendet wird, ist in den Informationen zur Anweisung CREATE TABLE dokumentiert.

Eine Inaktivierung der Wertkomprimierung hat folgende Auswirkungen:

- Die Klausel COMPRESS SYSTEM DEFAULTS wird implizit ebenfalls inaktiviert, sofern sie zuvor aktiviert war.
- Die unkomprimierten Spalten können dazu führen, dass die Zeilengröße das zulässige Maximum für die aktuelle Seitengröße des aktuellen Tabellenbereichs überschreitet. In diesem Fall wird die Fehlernachricht SQL0670N zurückgegeben.
- Vorhandene komprimierte Daten bleiben komprimiert, bis die Zeile aktualisiert wird oder Sie eine Tabellenreorganisation mit dem Befehl REORG ausführen.

Optimistisches Sperren - Übersicht

Durch die erweiterte Unterstützung für optimistisches Sperren wird ein Verfahren für SQL-Datenbankanwendungen bereitgestellt, bei dem Sperren zwischen dem Auswählen (SELECT) und dem Aktualisieren (UPDATE) bzw. Löschen (DELETE) von Zeilen nicht durchgehend beibehalten werden.

Anwendungen können so geschrieben werden, dass sie von der optimistischen Annahme ausgehen, dass es unwahrscheinlich ist, dass nicht gesperrte Zeilen vor dem Aktualisieren oder Löschen geändert werden. Wenn die Zeilen dennoch geändert werden, schlagen die Aktualisierungs- oder Löschoperationen fehl. In diesem Fall kann die Logik der Anwendung entsprechend reagieren, zum Beispiel indem sie eine erneute Auswahl (SELECT) versucht.

Der Vorteil dieses erweiterten optimistischen Sperrens liegt im besseren gemeinsamen Zugriff, da andere Anwendungen dieselben Zeilen lesen und schreiben können. Diese Technik des optimistischen Sperrens wird in dreischichtigen Umgebungen verwendet, in denen Geschäftsanwendungen keine Korrelation zu Datenbanktransaktionen besitzen, da Sperren nicht über Geschäftstransaktionen hinweg beibehalten werden können.

In Tabelle 23 sind die relevanten Themen für die einzelnen Kategorien aufgeführt.

Tabelle 23. Übersicht über die Informationen zum optimistischen Sperren

Kategorie	Zugehörige Themen
Allgemeine Informationen und Einschränkungen	• „Optimistisches Sperren" auf Seite 357 • „Granularität von Zeilenänderungstoken und falsche negative Werte" auf Seite 361 • „Einschränkungen und Aspekte des optimistischen Sperrens" auf Seite 359
Zeitbasierte Aktualisierungen	• „Zeitbasierte Aktualisierungserkennung" auf Seite 361 • „Für ROW CHANGE TIMESTAMP generierte Zeitwerte" auf Seite 363

Tabelle 23. Übersicht über die Informationen zum optimistischen Sperren (Forts.)

Kategorie	Zugehörige Themen
Aktivieren	• „Planen der Aktivierung des optimistischen Sperrens" auf Seite 365 • „Aktivieren des optimistischen Sperrens in Anwendungen" auf Seite 367
Verwendungsszenarios	• „Szenarios: Optimistisches Sperren und zeitbasierte Erkennung" auf Seite 441 – „Szenario: Verwenden des optimistischen Sperrens in einem Anwendungsprogramm" auf Seite 441 – „Szenario: Zeitbasierte Aktualisierungserkennung" auf Seite 445 – „Szenarios: Optimistisches Sperren mit implizit verdeckten Spalten" auf Seite 444
Referenzinformationen	• „Integrierte Funktion RID_BIT() und RID()" auf Seite 363 • ALTER TABLE (Anweisung) in *SQL Reference Volume 1* • CREATE TABLE (Anweisung) in *SQL Reference Volume 2* • DELETE (Anweisung) in *SQL Reference Volume 2* • SELECT (Anweisung) in *SQL Reference Volume 2* • UPDATE (Anweisung) in *SQL Reference Volume 2*

Anmerkung: In den Themen zum optimistischen Sperren ist jede Bezugnahme auf Zeilen, die eingefügt oder aktualisiert werden, so zu verstehen, dass diese Einfügung oder Aktualisierung das Ergebnis aller Formen von SQL-Anweisungen sein kann, die bewirken können, dass eine Zeile in eine Tabelle eingefügt oder in irgendeiner Weise aktualisiert wird. Zum Beispiel können die Anweisungen INSERT, UPDATE, MERGE und sogar DELETE (mit referenziellen Integritätsbedingungen) zur Folge haben, dass die Zeitmarkenspalte entweder erstellt oder aktualisiert wird.

Optimistisches Sperren

Beim *optimistischen Sperren* handelt es sich um ein Verfahren für SQL-Datenbankanwendungen, bei dem keine Zeilensperren zwischen dem Auswählen und dem Aktualisieren bzw. Löschen einer Zeile gehalten werden.

Die Anwendung wurde so geschrieben, dass sie von der optimistischen Annahme ausgeht, dass nicht gesperrte Zeilen vor der Aktualisierungs- oder Löschoperation wahrscheinlich nicht geändert werden. Wenn doch eine Zeile geändert wird, schlägt der Aktualisierungs- oder Löschvorgang fehl und die Anwendungslogik behandelt solche Fehler beispielsweise dadurch, dass sie die Auswahl wiederholt. Ein Vorteil des optimistischen Sperrens ist der verbesserte gemeinsame Zugriff, da andere Anwendungen diese Zeile lesen und schreiben können. In einer dreischichtigen Umgebung, in der Geschäftstransaktionen keine Korrelation mit Datenbanktransaktionen besitzen, wird das Verfahren des optimistischen Sperrens verwendet, da Sperren nicht über Geschäftstransaktionen hinweg beibehalten werden können.

Allerdings hat das optimistische Sperren nach Werten auch einige Nachteile:

- Es kann zu *falschen positiven Werten* ohne weitere Datenserverunterstützung führen, das heißt zu einer Bedingung bei Verwendung des optimistischen Sperrens, bei der eine Zeile, die seit ihrer Auswahl *geändert* wurde, nicht aktualisiert werden kann, ohne zunächst erneut ausgewählt zu werden. (Dies unterscheidet sich

von *falschen negativen Werten*, die eine Bedingung darstellen, bei der eine Zeile seit ihrer Auswahl *unverändert* geblieben ist, nicht aktualisiert werden kann, ohne zunächst erneut ausgewählt zu werden.)

- Es erfordert mehr Wiederholungslogik in Anwendungen.
- Für Anwendungen ist es schwierig, die UPDATE-Suchbedingungen zu bilden.
- Für den DB2-Server ist es ineffizient, nach der Zielzeile auf der Basis von Werten zu suchen.
- Datentypabweichungen zwischen einigen Clienttypen und Datenbanktypen, z. B. Zeitmarken, verhindern die Verwendung sämtlicher Spalten in der Aktualisierung mit Suche.

Die Unterstützung für ein einfacheres und schnelleres optimistisches Sperren ohne *falsche positive Werte* besitzt die folgenden neuen SQL-Funktionen, Ausdrücke und Komponenten:

- Integrierte Zeilenkennungsfunktion (RID_BIT oder RID)
- Ausdruck ROW CHANGE TOKEN
- Zeitbasierte Aktualisierungserkennung
- Implizit verdeckte Spalten

DB2-Anwendungen können das *optimistische Sperren nach Werten* ermöglichen, indem eine UPDATE-Anweisung mit Suche gebildet wird, durch die nach der Zeile mit genau denselben ausgewählten Werten gesucht wird. Die Anweisung UPDATE mit Suche schlägt fehl, wenn sich die Spaltenwerte der Zeile geändert haben.

Anwendungen, die dieses Programmiermodell verwenden, können von der verbesserten Funktion für optimistisches Sperren profitieren. Beachten Sie, dass Anwendungen, die dieses Programmiermodell nicht verwenden, nicht als Anwendungen für optimistisches Sperren betrachtet werden. Solche Anwendungen funktionieren weiterhin wie bisher.

Integrierte Zeilenkennungsfunktion (RID_BIT oder RID)
Diese integrierte Funktion kann in der SELECT-Liste oder in Vergleichselementen der Anweisung verwendet werden. In einem Vergleichselement (z. B. WHERE RID_BIT(tabelle)=?) wird das Gleichheitsvergleichselement mit RID_BIT als neue direkte Zugriffsmethode implementiert, um die Zeile effizient zu lokalisieren. Früher wurde eine VALUES-Methode zum optimistischen Sperren mit Werten angewendet, bei der alle ausgewählten Spaltenwerte den Vergleichselementen hinzugefügt wurden. Diese weniger effiziente Zugriffsmethode war davon abhängig, dass einige eindeutige Spaltenkombinationen jeweils nur eine Zeile kennzeichneten.

Ausdruck ROW CHANGE TOKEN
Dieser neue Ausdruck gibt ein Token vom Typ BIGINT zurück. Das Token stellt einen relativen Punkt in der Änderungsabfolge einer Zeile dar. Eine Anwendung kann den aktuellen Wert des Zeilenänderungstokens (ROW CHANGE TOKEN) einer Zeile mit dem Wert des Zeilenänderungstokens vergleichen, der beim letzten Abruf der Zeile gespeichert wurde, um zu ermitteln, ob die Zeile inzwischen geändert wurde.

Zeitbasierte Aktualisierungserkennung:
Diese Funktion wird SQL durch die Angaben RID_BIT() und ROW CHANGE TOKEN hinzugefügt. Zur Unterstützung dieser Funktion muss die Tabelle über eine neu generierte Spalte verfügen, die zum Speichern der Zeitmarkenwerte definiert wurde. Diese kann vorhandenen Tabellen mit der Anweisung ALTER TABLE hinzugefügt werden oder sie kann beim

Erstellen einer neuen Tabelle definiert werden. Das Vorhandensein der Spalte wirkt sich auch insofern auf das Verhalten des optimistischen Sperrens aus, als dass die Spalte, wenn sie zur Verbesserung der Granularität des Zeilenänderungstokens (ROW CHANGE TOKEN) von der Seitenebene auf die Zeilenebene verwendet wird, für Anwendungen mit optimistischem Sperren von großem Vorteil sein kann. Diese Funktion wurde auch DB2 für z/OS hinzugefügt.

Implizit verdeckte Spalten:
Aus Gründen der Kompatibilität vereinfacht diese Funktion die Verwendung der RID_BIT- und ROW CHANGE TOKEN-Spalten für vorhandene Tabellen und Anwendungen. Implizit verdeckte Spalten werden nicht extern zugänglich gemacht, wenn implizite Spaltenlisten verwendet werden. Beispiele:

- Eine Anweisung SELECT * für die Tabelle gibt keine implizit verdeckten Spalten in der Ergebnistabelle zurück.
- Eine Anweisung INSERT ohne Spaltenliste erwartet keinen Wert für implizit verdeckte Spalten. Allerdings muss die Spalte so definiert sein, dass sie Nullwerte zulässt oder einen anderen Standardwert hat.

Anmerkung: Im DB2-Glossar finden Sie die Definition von Termini zum optimistischen Sperren wie z. B. *optimistische Steuerung des gemeinsamen Zugriffs*, *pessimistisches Sperren*, *ROWID* und *Aktualisierungserkennung*.

Einschränkungen und Aspekte des optimistischen Sperrens

In diesem Abschnitt werden die Einschränkungen des optimistischen Sperrens aufgeführt, die zu beachten sind.

- Spalten für die Zeilenänderungszeitmarke (ROW CHANGE TIMESTAMP) werden in den folgenden Schlüsseln, Spalten und Namen nicht unterstützt (falls verwendet, wird SQLSTATE-Wert 429BV zurückgegeben):
  - Primärschlüssel
  - Fremdschlüssel
  - MDC-Spalten (MDC - Multidimensional Clustering)
  - Bereichspartitionierungsspalten
  - Hashpartitionierungsschlüssel für Datenbanken
  - Spalten mit der Integritätsbedingung DETERMINED BY
  - Kurznamen
- Die Funktion RID() wird in Konfigurationen mit partitionierten Datenbanken nicht unterstützt.
- Online- oder Offline-Reorganisationen von Tabellen, die zwischen den Abruf- und Aktualisierungsoperationen in einem Szenario mit optimistischem Sperren ausgeführt werden, können dazu führen, dass die Aktualisierung fehlschlägt. Dies sollte jedoch durch die normale Wiederholungslogik von Anwendungen aufgefangen werden.
- Das Attribut IMPLICITLY HIDDEN ist auf Spalten für die Zeilenänderungszeitmarke (ROW CHANGE TIMESTAMP) zu Zwecken des optimistischen Sperrens beschränkt.
- Inplace-Reorganisationen sind für Tabellen, wenn diesen vorhandenen Tabellen eine Spalte ROW CHANGE TIMESTAMP hinzugefügt wurde, eingeschränkt, bis für alle Zeilen garantiert ein Wert gespeichert wurde (bei diesem Fehler wird SQL2219 mit Ursachencode 13 zurückgegeben). Dies lässt sich mit dem Befehl **LOAD REPLACE** bzw. mit einer klassischen Tabellenreorganisation erreichen, sodass

falsche Positivwerte verhindert werden. Tabellen, die bereits mit der Spalte ROW CHANGE TIMESTAMP erstellt werden, haben keine Einschränkungen.

Aspekte für implizit verdeckte Spalten

Eine mit IMPLICITLY HIDDEN definierte Spalte gehört nicht zur Ergebnistabelle einer Abfrage, in der eine SELECT-Liste mit der Notation '*' angegeben wird. Dennoch kann eine implizit verdeckte Spalte in einer Abfrage explizit angegeben werden.

Wenn in der Einfügung keine Spaltenliste angegeben wird, darf die Klausel VALUES oder die SELECT-Liste für die Einfügung diese Spalte nicht enthalten (im Allgemeinen muss es sich bei dieser Spalte um eine generierte Spalte handeln, die einen Standardwert hat oder die Nullwerte enthalten darf).

Eine implizit verdeckte Spalte kann zum Beispiel in der SELECT-Liste oder in einem Vergleichselement einer Abfrage angegeben werden. Darüber hinaus kann eine implizit verdeckte Spalte auch in einer Anweisung CREATE INDEX, ALTER TABLE, INSERT, MERGE oder UPDATE explizit angegeben werden. Eine implizit verdeckte Spalte kann in einer referenziellen Integritätsbedingung angegeben werden. Eine Klausel REFERENCES, die keine Spaltenliste enthält, bezieht sich implizit auf den Primärschlüssel der übergeordneten Tabelle. Es ist möglich, dass der Primärschlüssel der übergeordneten Tabelle eine Spalte enthält, die als implizit verdeckt definiert ist. Eine solche referenzielle Integritätsbedingung ist zulässig.

- Wenn die SELECT-Liste des Fullselects der MQT-Definition explizit eine implizit verdeckte Spalte angibt, wird diese Spalte in die MQT (Materialized Query Table, gespeicherte Abfragetabelle) eingefügt. Ansonsten ist eine implizit verdeckte Spalte nicht Teil einer MQT, die sich auf eine Tabelle bezieht, die eine implizit verdeckte Spalte enthält.
- Wenn die SELECT-Liste des Fullselects einer Sichtdefinition (Anweisung CREATE VIEW) explizit eine implizit verdeckte Spalte angibt, wird diese Spalte in die Sicht eingefügt (jedoch wird die Spalte in der Sicht nicht als verdeckt betrachtet). Ansonsten ist eine implizit verdeckte Spalte nicht Teil einer Sicht, die sich auf eine Tabelle bezieht, die eine implizit verdeckte Spalte enthält.

Aspekte für die kennsatzbasierte Zugriffssteuerung (LBAC)

Wenn eine Spalte durch die kennsatzbasierte Zugriffssteuerung (LBAC - Label Based Access Control) geschützt wird, wird der Zugriff eines Benutzers auf diese Spalte durch die LBAC-Richtlinien und den Sicherheitskennsatz des Benutzers bestimmt. Dieser Schutz erstreckt sich, sofern er sich auf eine Spalte für die Zeilenänderungszeitmarke bezieht, auf den Verweis auf diese Spalte durch die Ausdrücke ROW CHANGE TIMESTAMP und ROW CHANGE TOKEN, die aus dieser Spalte abgeleitet werden.

Bei der Festlegung der Sicherheitsrichtlinien für eine Tabelle muss daher sichergestellt werden, dass der Zugriff auf die Spalte für die Zeilenänderungszeitmarke (ROW CHANGE TIMESTAMP) für alle Benutzer verfügbar ist, die mit dem optimistischen Sperren bzw. der zeitbasierten Aktualisierungserkennung arbeiten müssen. Beachten Sie, dass bei einer fehlenden Spalte für die Zeilenänderungszeitmarke der Ausdruck ROW CHANGE TOKEN nicht durch LBAC blockiert werden kann. Wenn die Tabelle jedoch geändert wird, um eine Spalte für die Zeilenänderungszeitmarke hinzuzufügen, gelten alle Aspekte der kennsatzbasierten Zugriffssteuerung (LBAC).

Granularität von Zeilenänderungstoken und falsche negative Werte

Die integrierte Funktion RID_BIT() und das Zeilenänderungstoken sind die einzigen Anforderungen für das optimistische Sperren. Allerdings wirkt sich das Schema der Tabelle auch auf das Verhalten des optimistischen Sperrens aus.

Beispiel: Eine Zeitmarkenspalte für Zeilenänderung, die so definiert wurde, dass sie eine der folgenden Anweisungsklauseln verwendet, veranlasst den DB2-Server zur Speicherung der Zeit der letzten Änderung (oder ersten Einfügung) einer Zeile. Dies ist eine Möglichkeit, die Zeitmarke der allerletzten Änderung einer Zeile zu erfassen. Dabei handelt es sich um eine Zeitmarkenspalte, die vom Datenbankmanager verwaltet wird, es sei denn, die Klausel GENERATED BY DEFAULT wird zum Akzeptieren eines vom Benutzer bereitgestellten Eingabewerts verwendet.

```
GENERATED ALWAYS FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP
GENERATED BY DEFAULT FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP
```

Wenn eine Anwendung also den neuen Ausdruck ROW CHANGE TOKEN in einer Tabelle verwendet, sind die folgenden beiden Möglichkeiten zu beachten:

- *Die Tabelle verfügt über keine Zeitmarkenspalte für Zeilenänderung:* Der Ausdruck ROW CHANGE TOKEN gibt den abgeleiteten Wert BIGINT zurück, der von allen Zeilen gemeinsam genutzt wird, die sich auf derselben Seite befinden. Wenn eine einzige Zeile auf einer Seite aktualisiert wird, wird das Zeilenänderungstoken (ROW CHANGE TOKEN) für alle Zeilen auf derselben Seite geändert. Dies bedeutet, dass eine Aktualisierung fehlschlagen kann, wenn Änderungen an anderen Zeilen vorgenommen werden. Diese Eigenschaft wird als 'falscher negativer Wert' bezeichnet.

  Anmerkung: Verwenden Sie diesen Modus nur, wenn die Anwendung *falsche negative Werte* tolerieren kann und nicht jeder Zeile für eine Zeitmarkenspalte für Zeilenänderung (ROW CHANGE TIMESTAMP) zusätzlichen Speicher hinzufügen will.
- *Die Tabelle verfügt über eine Zeitmarkenspalte für Zeilenänderung:* Der Ausdruck ROW CHANGE TOKEN gibt den Wert BIGINT zurück, der von dem Zeitmarkenwert in der Spalte abgeleitet wird. In diesem Fall können *falsche negative Werte* auftreten, jedoch relativ selten: Wenn die Tabelle reorganisiert oder umverteilt wird, können *falsche negative Werte* auftreten, wenn die Zeile versetzt wird und eine Anwendung den früheren Wert von RID_BIT() verwendet.

Zeitbasierte Aktualisierungserkennung

Für einige Anwendungen sind Datenbankaktualisierungen für bestimmte Zeitbereiche erforderlich, die z. B. zur Datenreplikation oder für Prüfungsszenarios verwendet werden. Diese Informationen werden vom Ausdruck ROW CHANGE TIMESTAMP bereitgestellt.

```
ROW CHANGE TIMESTAMP FOR tabellenbezeichnung
```

Dieser Ausdruck gibt eine Zeitmarke zurück, die die Zeit der letzten Änderung einer Zeile darstellt. Diese Zeit wird als Ortszeit ähnlich wie bei CURRENT TIMESTAMP angegeben. Für eine aktualisierte Zeile wird dadurch die letzte Aktualisierung der Zeile wiedergegeben. Ansonsten entspricht der Wert der ursprünglichen Einfügung der Zeile.

Der Wert des Ausdrucks ROW CHANGE TIMESTAMP unterscheidet sich von der aktuellen Zeitmarke (CURRENT TIMESTAMP) insofern, als dass sie garantiert eindeutig ist, wenn sie von der Datenbank pro Zeile pro Datenbankpartition zugewiesen wird. Es handelt sich um eine lokale Zeitmarkennäherung des Änderungszeit-

punkts jeder einzelnen Zeile, die eingefügt oder aktualisiert wird. Da der Wert stets von früher zu später wächst, kann er unter folgenden Umständen vom Wert der Systemuhr abweichen:

- Wenn die Systemuhr geändert wurde.
- Wenn die Zeitmarkenspalte für Zeilenänderung mit GENERATED BY DEFAULT definiert ist (nur für die Datenweitergabe gedacht) und eine Zeile mit einem Wert bereitgestellt wird, der nicht synchron ist.

Als Voraussetzung für die Verwendung des Ausdrucks ROW CHANGE TIMESTAMP muss die Tabelle über eine definierte Zeitmarkenspalte für Zeilenänderung verfügen, die die Standardgenauigkeit für den Zeitmarkendatentyp TIMESTAMP(6) (oder TIMESTAMP - Standardgenauigkeit ist 6) verwendet. Jede Zeile gibt die Zeitmarke ihrer Einfügung oder ihrer letzten Aktualisierung zurück. Es gibt zwei Methoden, durch die die Zeitmarkenspalte für Zeilenänderung Teil der Tabelle sein kann:

- Die Tabelle wurde mit der Klausel FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP der Anweisung CREATE TABLE erstellt. Der Ausdruck ROW CHANGE TIMESTAMP gibt den Wert der Spalte zurück. Für diese Kategorie ist die Zeitmarke genau. Allgemein gilt, dass die Zeitmarke für Zeilenänderung, wenn sie durch die Datenbank generiert wird, von der Geschwindigkeit von Einfügeoperationen sowie von möglichen Systemuhrumstellungen, zum Beispiel aufgrund der Sommerzeit, beeinflusst wird.
- Die Tabelle wurde nicht mit einer Zeitmarkenspalte für die Zeilenänderung erstellt, sondern eine solche Spalte wurde später mit der Klausel FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP einer Anweisung ALTER TABLE hinzugefügt. Der Ausdruck ROW CHANGE TIMESTAMP gibt den Wert der Spalte zurück. Bei dieser Kategorie erhalten die alten Zeilen (d. h. die Zeilen vor der ALTER-Operation) die tatsächliche Zeitmarke erst, wenn sie das erste Mal aktualisiert werden oder eine Offlinetabellenreorganisation durchgeführt wird.

  Anmerkung: Diese Zeitmarke ist ein ungefährer Wert für den Zeitpunkt, zu dem die tatsächliche Aktualisierung in der Datenbank stattfand. Dieser Wert basiert auf der Systemuhr zum jeweiligen Zeitpunkt und berücksichtigt die Einschränkung, dass keine Zeitmarke innerhalb einer Datenbank- bzw. Tabellenpartition wiederholt werden kann. In der Praxis ist dies in der Regel eine sehr genaue Darstellung der Zeit der Aktualisierung. Allgemein gilt, dass die Zeitmarke für Zeilenänderung, wenn sie durch die Datenbank generiert wird, von der Geschwindigkeit von Einfügeoperationen sowie von möglichen Systemuhrumstellungen, zum Beispiel aufgrund der Sommerzeit, beeinflusst wird.

  Zeilen, die seit der Anweisung ALTER TABLE nicht aktualisiert wurden, geben den Standardwert für den Typ der Spalte zurück. Dieser Standardwert ist Mitternacht, 01. Januar, Jahr 1. Nur Zeilen, die aktualisiert wurden, haben eine eindeutige Zeitmarke. Zeilen, bei denen die Zeitmarke über eine Offlinetabellenreorganisation gespeichert wird, geben eine eindeutige Zeitmarke zurück, die bei der Reorganisation der Tabelle generiert wird. Eine Operation **REORG TABLE** mit dem Parameter **INPLACE** genügt nicht, da Schemaänderungen damit nicht gespeichert werden.

In beiden Fällen kann die Zeitmarke einer Zeile auch aktualisiert werden, wenn eine Umverteilung durchgeführt wird. Wenn die Zeile bei der Umverteilung von einer Datenbankpartition in eine andere versetzt wird, muss eine neue Zeitmarke generiert werden, die für das Ziel garantiert eindeutig ist.

Für ROW CHANGE TIMESTAMP generierte Zeitwerte

Es gibt einige Grenzwertbedingungen zu den genauen Werten, die für die Zeitmarkenspalten für Zeilenänderung aufgrund der Umsetzung eindeutiger Werte pro Partition generiert werden.

Wenn die Systemuhr in der Vergangenheit wegen Uhrzeitfehlern oder einer Sommerzeitrichtlinie auf dem DB2-Server angepasst wird, werden Zeitmarken möglicherweise in der Zukunft relativ zum aktuellen Wert der Systemuhr oder dem Wert des Sonderregisters CURRENT TIMESTAMP angezeigt. Dazu kommt es, wenn eine Zeitmarke vor der Anpassung der Systemuhr generiert wurde, d. h., später als die angepasste Zeit, da die Zeitmarken zur Wahrung der Eindeutigkeit immer aufsteigend generiert werden.

Wenn Zeitmarken für Spalten generiert werden, die der Tabelle durch eine **REORG TABLE**-Operation oder im Rahmen einer **LOAD**-Operation hinzugefügt wurden, werden die Zeitmarken an einem Punkt während der Verarbeitung des Dienstprogramms nacheinander generiert; dabei wird bei einem Anfangszeitmarkenwert begonnen. Wenn das Dienstprogramm Zeilen schneller als die Zeitmarkengranularität (mehr als 1 Million Zeilen pro Sekunde) verarbeiten kann, können die für einige Zeilen generierten Werte relativ zur Systemuhr oder dem Sonderregister CURRENT TIMESTAMP scheinbar in der Zukunft liegen.

In jedem Fall gibt es eine enge Annäherung an die Zeit, zu der die Zeile eingefügt wurde, wenn die Systemuhr dieselben Werte für Zeitmarkenspalten für Zeilenänderung hat. Bis zu diesem Zeitpunkt werden Zeitmarken in aufsteigender Reihenfolge auf der Basis der feinsten Granularität, die für den Zeitmarkentyp timestamp(6) zulässig ist, generiert.

Integrierte Funktion RID_BIT() und RID()

Die Funktion RID_BIT() und das Zeilenänderungstoken (ROW CHANCE TOKEN) können für jede Zeile in einer Tabelle ausgewählt werden. Die SELECT-Operation kann auf einer beliebigen Isolationsstufe erfolgen, die für die Anwendung erforderlich ist.

Die Anwendung kann dieselbe (ungeänderte) Zeile mit optimistischem Sperren ändern, indem sie unter Verwendung der beiden folgenden Elemente sucht:

- Die Funktion RID_BIT(), um direkt auf die zu Zielzeile zuzugreifen (ohne Suche)
- Das Zeilenänderungstoken (ROW CHANGE TOKEN), um sicherzustellen, dass es sich um dieselbe ungeänderte Zeile handelt

Diese Aktualisierung (oder Löschung) kann zu einem beliebigen Zeitpunkt nach der Auswahl (SELECT) innerhalb derselben UOW (Unit of Work, Arbeitseinheit) erfolgen oder sogar über Verbindungsgrenzen hinweg. Die einzige Voraussetzung besteht darin, dass die beiden oben genannten Werte für eine bestimmte Zeile zu einem früheren Zeitpunkt abgerufen worden sein müssen.

Die Technik des optimistischen Sperrens wird im „WebSphere-Oriented Programming Model" verwendet. Zum Beispiel verwendet Microsoft .NET dieses Modell, um SELECT-Anweisungen, auf die UPDATE- oder DELETE-Anweisungen folgen, wie folgt zu verarbeiten:

- Die Verbindung zum Datenbankserver wird hergestellt und die gewünschten Zeilen werden per SELECT aus einer Tabelle ausgewählt.
- Trennen Sie die Verbindung zur Datenbank oder geben Sie die Zeilensperren frei, damit andere Anwendungen Daten lesen, aktualisieren, löschen und einfügen können, ohne beim gemeinsamen Zugriff auf Konflikte aufgrund von Sper-

ren und Ressourcen zu treffen, die von der Anwendung genutzt werden. Die Isolationsstufe UR („Uncommited Read" - nicht festgeschriebener Lesevorgang) lässt einen höheren gemeinsamen Zugriff zu und unter der Annahme, dass andere Anwendungen ihre Aktualisierungs- und Löschtransaktionen festschreiben (COMMIT), liest diese Anwendung mit optimistischem Sperren die aktualisierten Werte und die optimistisch gesuchte Aktualisierung bzw. Löschung wird erfolgreich ausgeführt.

- Es werden einige lokale Berechnungen an den ausgewählten Zeilendaten ausgeführt.
- Die Verbindung zum Datenbankserver wird erneut hergestellt und es erfolgt eine Suche in einer oder mehreren Zielzeilen für die UPDATE- oder DELETE-Operation. (Wenn die Zielzeile geändert wurde, schlagen die UPDATE- oder DELETE-Anweisungen fehl und werden entsprechend verarbeitet).

Anwendungen, die dieses Programmiermodell verwenden, können von der verbesserten Funktion für optimistisches Sperren profitieren. Beachten Sie, dass Anwendungen, die dieses Programmiermodell nicht verwenden, nicht als Anwendungen für optimistisches Sperren betrachtet werden. Solche Anwendungen funktionieren weiterhin wie bisher.

Merkmale der integrierten Funktion RID_BIT() und RID()

Die folgenden neuen Funktionen werden zum Zweck eines verbesserten optimistischen Sperrens und der Aktualisierungserkennung implementiert:

RID_BIT(*tabellenbezeichnung* **)**
Eine neue integrierte Funktion, die die Satzkennung (RID, Record Identifier) einer Zeile mit dem Datentyp VARCHAR(16) FOR BIT DATA zurückgibt.

Anmerkung: DB2 for z/OS implementiert eine integrierte Funktion RID mit dem Rückgabetyp BIGINT, der jedoch für Linux-, UNIX- und Windows-RIDs nicht groß genug ist. Aus Kompatibilitätsgründen gibt diese integrierte Funktion RID() ebenso wie die Funktion RID_BIT() Daten des Typs BIGINT zurück.

Die integrierte Funktion RID() funktioniert nicht in Umgebungen mit partitionierten Datenbanken und liefert keine Tabellenversionsinformationen. Ansonsten funktioniert sie in der gleichen Weise wie die Funktion RID_BIT. Verwenden Sie diese Funktion nur zur Codierung von Anwendungen, die auf z/OS-Server portiert werden sollen. Die Ausführungen dieses Abschnitts beziehen sich nur auf die Funktion RID_BIT, sofern nichts anderes erforderlich ist.

Integrierte Funktion RID_BIT()
Diese integrierte Funktion kann in der Anweisung der SELECT-Liste oder der Vergleichselemente verwendet werden. In einem Vergleichselement (z. B. `WHERE RID_BIT(tabelle)=?`) wird das Gleichheitsvergleichselement mit RID_BIT als neue direkte Zugriffsmethode implementiert, um die Zeile effizient zu lokalisieren. Früher wurde eine VALUES-Methode zum *optimistischen Sperren mit Werten* angewendet, bei der alle ausgewählten Spaltenwerte den Vergleichselementen hinzugefügt wurden. Diese weniger effiziente Zugriffsmethode war davon abhängig, dass einige eindeutige Spaltenkombinationen jeweils nur eine Zeile kennzeichneten.

ROW CHANGE TOKEN FOR *tabellenbezeichnung*
Ein neuer Ausdruck, der ein Token vom Typ BIGINT zurückgibt. Das To-

ken stellt einen relativen Punkt in der Änderungsabfolge einer Zeile dar. Eine Anwendung kann den aktuellen Wert des Zeilenänderungstokens (ROW CHANGE TOKEN) einer Zeile mit dem Wert des Zeilenänderungstokens vergleichen, der beim letzten Abruf der Zeile gespeichert wurde, um zu ermitteln, ob die Zeile inzwischen geändert wurde.

Spalte ROW CHANGE TIMESTAMP
Eine generierte Spalte (GENERATED) mit dem Standarddatentyp TIMESTAMP, die wie folgt definiert werden kann:

```
GENERATED ALWAYS FOR EACH ROW ON UPDATE
AS ROW CHANGE TIMESTAMP
```

oder (nur für Operationen zur Datenweitergabe bzw. zum Entladen und erneuten Laden empfohlen):

```
GENERATED BY DEFAULT FOR EACH ROW ON UPDATE
AS ROW CHANGE TIMESTAMP
```

Die Daten in dieser Spalte ändern sich bei jeder Änderung der Zeile. Wenn diese Spalte definiert ist, wird der ROW CHANGE TOKEN-Wert aus dieser Spalte abgeleitet. Beachten Sie, dass der Datenbankmanager bei Angabe von GENERATED ALWAYS dafür sorgt, dass dieser Wert innerhalb der Datenbankpartition bzw. der Tabellenpartition eindeutig ist, um sicherzustellen, dass keine *falschen Positivwerte* zugelassen werden.

Zur Verwendung der ersten beiden Elemente RID_BIT und ROW CHANGE TOKEN sind keine weiteren Änderungen am Datenbankschema erforderlich. Beachten Sie jedoch, dass ohne die Spalte ROW CHANGE TIMESTAMP das Zeilenänderungstoken ROW CHANGE TOKEN von allen Zeilen auf der gleichen Seite gemeinsam verwendet wird. Aktualisierungen an einer Zeile auf der Seite können *falsche Negativwerte* für andere, auf derselben Seite gespeicherten Zeilen zur Folge haben. Mit dieser Spalte wird das Zeilenänderungstoken (ROW CHANGE TOKEN) aus der Zeitmarke abgeleitet und von keinen anderen Zeilen in der Tabelle bzw. der Datenbankpartition mit verwendet. Siehe „Granularität von Zeilenänderungstoken und falsche negative Werte" auf Seite 361.

Zeitbasierte Aktualisierungserkennung und die Funktionen RID_BIT() und RID()

Der Ausdruck ROW CHANGE TIMESTAMP gibt einen Zeitmarkenwert zurück, der den Zeitpunkt darstellt, zu dem die Zeile in der durch die Tabellenbezeichnung angegebenen Tabelle zuletzt geändert wurde. Trotz der gegenseitigen Beziehung der integrierten Funktion RID_BIT() bzw. RID() und der Funktion zur zeitbasierten Aktualisierungserkennung, ist es wichtig zu beachten, dass die Ausdrücke ROW CHANGE TOKEN und ROW CHANGE TIMESTAMP nicht gegenseitig austauschbar verwendet werden können. Insbesondere ist zu beachten, dass der Ausdruck ROW CHANGE TIMESTAMP kein Bestandteil der Verwendung des optimistischen Sperrens ist.

Planen der Aktivierung des optimistischen Sperrens

Seitdem sich die neuen SQL-Ausdrücke und Attribute für optimistisches Sperren ohne DDL-Änderungen an den Tabellen verwenden lassen, können Sie das optimistische Sperren problemlos in Ihren Testanwendungen ausprobieren.

Beachten Sie, dass Anwendungen mit optimistischem Sperren ohne DDL-Änderungen möglicherweise mehr *falsche Negativwerte* empfangen als mit DDL-Änderungen. Eine Anwendung, die falsche Negativwerte empfängt, lässt sich in einer Produktionsumgebung eventuell nicht gut skalieren, weil die falschen Negativwerte

zu viele Wiederholungsversuche verursachen. Zur Vermeidung falscher Negativwerte sollten Zieltabellen mit optimistischem Sperren wie folgt behandelt werden:

- Sie sollten mit einer Spalte ROW CHANGE TIMESTAMP erstellt werden.
- Sie sollten geändert werden, sodass sie eine Spalte ROW CHANGE TIMESTAMP enthalten.

Wenn die empfohlenen DDL-Änderungen ausgeführt wurden, werden falsche Negativwerte nur noch selten auftreten. Die einzigen falschen Negativwerte, die möglicherweise auftreten, werden die Folge von Operationen auf Tabellenebene sein, wie zum Beispiel eine Reorganisation (REORG), bei der nicht gleichzeitig ausgeführte Anwendungen verschiedene Zeilen bearbeiten.

Im Allgemeinen lässt der Datenbankmanager falsche Negativwerte (z. B. bei Online- oder Offline-Reorganisation) zu, und das Vorhandensein einer Spalte für die Zeilenänderungszeitmarke (ROW CHANGE TIMESTAMP) ist ausreichend, um zu bestimmen, ob die Granularität auf Seiten- oder Zeilenebene verwendet wird. Sie können auch SYSCAT.COLUMNS nach einer Tabelle abfragen, die Zeilen mit dem Wert YES in der Spalte ROWCHANGETIMESTAMP enthält.

Eine gründliche Analyse der Anwendung und der Datenbank kann möglicherweise zu dem Ergebnis kommen, dass diese DDL-Anweisungen nicht erforderlich sind, wenn zum Beispiel nur eine Zeile pro Seite vorhanden ist oder wenn die Aktualisierungs- und Löschoperationen nur sporadisch und selten oder auch nie an derselben Datenseite ausgeführt werden. Eine solche Analyse ist jedoch die Ausnahme.

Für den Zweck der Erkennung von Aktualisierungszeitmarken müssen Sie Änderungen an den DDL-Anweisungen für die Tabelle vornehmen und möglicherweise die Tabelle reorganisieren, um die Werte real zu speichern. Wenn Sorge besteht, dass diese Änderungen negative Auswirkungen auf die Produktionsdatenbank haben könnten, sollten die Änderungen zunächst an Prototypen in einer Testumgebung getestet werden. Zum Beispiel können sich die zusätzlichen Spalten auf die Zeilengrößenbegrenzen und die Planauswahl auswirken.

Zu beachtende Bedingungen

- Sie sollten sich über Bedingungen im Klaren sein, die die Systemuhr und die Granularität der Zeitmarkenwerte betreffen. Wenn eine Tabelle eine Spalte ROW CHANGE TIMESTAMP besitzt, enthält die neue Zeile nach einer Einfügung oder Aktualisierung einen eindeutigen ROW CHANGE TIMESTAMP-Wert in dieser Tabelle in dieser Datenbankpartition.
- Zur Sicherstellung der Eindeutigkeit wird die generierte Zeitmarke einer Zeile immer erhöht, unabhängig davon, ob die Systemuhr zurückgestellt wird oder ob die Aktualisierung bzw. Einfügung von Daten schneller erfolgt, als dies durch die Granularität der Zeitmarke erfasst wird. Daher kann ROW CHANGE TIMESTAMP in Relation zur Systemzeit und zum Sonderregister CURRENT TIMESTAMP einen Wert in der Zukunft enthalten. Sofern die Systemuhr nicht vollkommen aus der Synchronisation gerät oder der Datenbankmanager nicht mehr als eine Million Zeilen pro Sekunde einfügt oder aktualisiert, sollte dieser Wert in der Regel sehr nahe an der tatsächlichen Zeit liegen. Im Unterschied zum Wert des Sonderregisters CURRENT TIMESTAMP wird dieser Wert außerdem pro Zeile zum Zeitpunkt der Aktualisierung generiert, sodass er normalerweise wesentlich näher an der Realzeit liegt als der Wert in CURRENT TIMESTAMP, der nur einmal für die gesamte Anweisung generiert wird, deren Ausführung je nach Komplexität und Anzahl der betroffenen Zeilen geraume Zeit in Anspruch nehmen kann.

Aktivieren des optimistischen Sperrens in Anwendungen

Zur Aktivierung der Unterstützung für optimistisches Sperren in Anwendungen müssen Sie eine Reihe von Schritten ausführen.

Vorgehensweise

1. Rufen Sie in der einleitenden Abfrage mit SELECT die Satz-ID (RID, siehe „Integrierte Funktion RID_BIT() und RID()" auf Seite 363) und das Zeilenänderungstoken (ROW CHANGE TOKEN) für jede Zeile ab, die Sie verarbeiten müssen.
2. Geben Sie die Zeilensperren frei, sodass andere Anwendungen SELECT-, INSERT-, UPDATE- und DELETE-Anweisungen an der Tabelle ausführen können.
3. Führen Sie eine UPDATE- oder DELETE-Anweisung durch eine gezielte Suche mit der Satz-ID und dem Zeilenänderungstoken (ROW CHANGE TOKEN) in der Suchbedingung an den Zielzeilen aus, wobei Sie optimistisch annehmen, dass die nicht gesperrte Zeile seit Ausführung der ursprünglichen SELECT-Anweisung nicht geändert wurde.
4. Falls die Zeile geändert wurde, schlägt die UPDATE-Operation fehl und die Anwendungslogik muss den Fehler behandeln. Die Anwendung kann zum Beispiel die SELECT- und die UPDATE-Operation wiederholen.

Nächste Schritte

Nach der Ausführung der obigen Schritte:

- Wenn die Anzahl der von Ihrer Anwendung ausgeführten Wiederholversuche höher scheint als erwartet oder als gewünscht wird, können Sie Ihrer Tabelle eine Spalte für die Zeilenänderungszeitmarke (ROW CHANGE TIMESTAMP) hinzufügen, um sicherzustellen, dass nur Änderungen an der Zeile, die von der Funktion RID_BIT ermittelt wird, nur das Zeilenänderungstoken (ROW CHANGE TOKEN) ungültig machen und keine anderen Aktivitäten an derselben Seite ausführen.
- Zum Anzeigen von Zeilen, die innerhalb eines bestimmten Zeitrahmens eingefügt oder aktualisiert wurden, erstellen oder ändern Sie die Tabelle in der Weise, dass sie eine Spalte für die Zeilenänderungszeitmarke (ROW CHANGE TIMESTAMP) enthält. Diese Spalte wird automatisch vom Datenbankmanager gepflegt und kann entweder über den Spaltennamen oder mit dem Ausdruck ROW CHANGE TIMESTAMP abgefragt werden.
- Nur für Spalten der Zeilenänderungszeitmarke gilt, dass die Spalte, wenn sie mit dem Attribut IMPLICITLY HIDDEN als implizit verdeckt definiert wurde, nicht extern bereitgestellt wird, wenn ein impliziter Verweis auf die Spalten der Tabelle erfolgt. Eine implizit verdeckte Spalte kann in SQL-Anweisungen jedoch immer explizit angegeben werden. Dies kann nützlich sein, wenn eine Spalte, die einer Tabelle hinzugefügt wird, dazu führt, dass vorhandene Anwendungen mit impliziten Spaltenlisten fehlschlagen.

Tabellenpartitionierung und Datenorganisationsschemata

Die Tabellenpartitionierung ist ein Datenorganisationsschema, bei dem Tabellendaten auf mehrere Datenpartitionen entsprechend den Werten einer oder mehrerer Partitionierungsspalten der Tabelle verteilt werden. Daten aus einer Tabelle werden in mehrere Speicherobjekte partitioniert, die sich in verschiedenen Tabellenbereichen befinden können.

Vollständige und detaillierte Informationen zur Tabellenpartitionierung und zu Datenorganisationsschemata finden Sie im Handbuch *Partitionierung und Clustering*.

Erstellen von Tabellen

Der Datenbankmanager steuert den Zugriff auf die in den Tabellen gespeicherten Daten und deren Änderungen. Tabellen können mit der Anweisung CREATE TABLE erstellt werden.

Mithilfe komplexer Anweisungen können alle Attribute und Merkmale von Tabellen definiert werden. Wenn jedoch alle Standardwerte genutzt werden, ist die Anweisung zur Erstellung einer Tabelle einfach.

Deklarieren temporärer Tabellen

Zur Definition temporärer Tabellen aus Anwendungen heraus verwenden Sie die Anweisung DECLARE GLOBAL TEMPORARY TABLE.

Informationen zu diesem Vorgang

Temporäre Tabellen, die auch als benutzerdefinierte temporäre Tabellen bezeichnet werden, werden von Anwendungen verwendet, die mit Daten in der Datenbank arbeiten. Ergebnisse aus der Bearbeitung der Daten müssen temporär in einer Tabelle gespeichert werden. Ein Tabellenbereich für temporäre Benutzertabellen muss vorhanden sein, bevor temporäre Tabellen deklariert werden.

Anmerkung: Die Beschreibung temporärer Tabellen wird im Systemkatalog nicht aufgeführt. Dadurch sind sie nicht persistent für andere Anwendungen und können nicht mit anderen Anwendungen gemeinsam genutzt werden. Wenn die Anwendung, die diese Tabelle verwendet, beendet wird oder die Verbindung zur Datenbank trennt, werden alle Daten in dieser Tabelle gelöscht und auch die Tabelle selbst wird implizit gelöscht.
Folgende Spalten werden von temporären Tabellen nicht unterstützt:

- Spalten mit benutzerdefinierten Datentypen
- LONG VARCHAR-Spalten
- XML-Spalten für erstellte globale temporäre Tabellen

Beispiel

```
DECLARE GLOBAL TEMPORARY TABLE temptbl
   LIKE empltabl
   ON COMMIT DELETE ROWS
   NOT LOGGED
   IN usr_tbsp
```

Diese Anweisung definiert eine temporäre Tabelle mit dem Namen `temptbl`. Diese Tabelle wird mit Spalten definiert, die exakt dieselben Namen und Beschreibungen wie die Spalten der Tabelle `empltabl` haben. Die implizite Definition umfasst nur den Spaltennamen, den Datentyp, das Merkmal für die Optionalität der Dateneingabe sowie die Standardwertattribute für die Spalte. Alle anderen Spaltenattribute, einschließlich der eindeutigen Integritätsbedingungen, der Integritätsbedingungen über Fremdschlüssel sowie der Trigger und Indizes, werden nicht definiert. Durch die Option ON COMMIT DELETE ROWS (jede DELETE ROWS-Option) löscht der Datenbankmanager Zeilen immer, unabhängig davon, ob ein WITH HOLD-Cursor für die Tabelle geöffnet ist oder nicht. Der Datenbankmanager optimiert eine Löschoperation mit NOT LOGGED, indem er eine interne TRUNCATE-Operation implementiert, wenn keine WITH HOLD-Cursor geöffnet sind. Ansonsten löscht der Datenbankmanager die Zeilen jeweils einzeln.

Die Tabelle wird implizit gelöscht, wenn die Anwendung die Verbindung zur Datenbank trennt. Weitere Informationen finden Sie in der Beschreibung der Anweisung DECLARE GLOBAL TEMPORARY TABLE.

Erstellen von temporären Tabellen und Herstellen von Verbindungen zu den erstellten temporären Tabellen

Erstellte temporäre Tabellen werden mit der Anweisung CREATE GLOBAL TEMPORARY TABLE erstellt. Wenn eine Anwendung zum ersten Mal über eine Verbindung auf eine erstellte temporäre Tabelle verweist, wird eine private Version der erstellten temporären Tabelle zur Verwendung durch die Anwendung über die Verbindung instanziiert.

Informationen zu diesem Vorgang

Ähnlich wie deklarierte temporäre Tabellen werden erstellte temporäre Tabellen von Anwendungen verwendet, die mit Daten in der Datenbank arbeiten und die Ergebnisse der Bearbeitung der Daten temporär in einer Tabelle speichern müssen. Während die Informationen einer deklarierten temporären Tabelle *nicht* in den Systemkatalogtabellen gespeichert werden, sodass sie in jeder Sitzung, in der sie verwendet wird, definiert werden muss, *werden* die Informationen einer erstellten temporären Tabelle im Systemkatalog gespeichert, sodass es nicht erforderlich ist, sie in jeder Sitzung, in der sie verwendet wird, zu definieren. Das heißt, sie ist persistent und kann mit anderen Anwendungen über verschiedene Verbindungen hinweg gemeinsam genutzt werden. Ein Tabellenbereich für temporäre Benutzertabellen muss vorhanden sein, bevor erstellte temporäre Tabellen erstellt werden können.

Anmerkung: Der erste implizite oder explizite Verweis auf die erstellte temporäre Tabelle, der von einem Programm ausgeführt wird, das die Verbindung verwendet, erstellt eine leere Instanz der jeweiligen erstellten temporären Tabelle. Jede Verbindung, die auf diese erstellte temporäre Tabelle verweist, hat eine eigene, für sich spezifische Instanz der erstellten temporären Tabelle, und die Instanz bleibt nicht über die Dauer der Verbindung hinaus bestehen.

Verweise auf den Namen einer erstellten temporären Tabelle in mehreren Verbindungen beziehen sich auf dieselbe eine persistente Definition für die erstellte temporäre Tabelle, jedoch auf eine jeweils andere Instanz der erstellten temporären Tabelle für jede Verbindung auf dem aktuellen Server. Wenn der Name der erstellten temporären Tabelle, auf den verwiesen wird, nicht qualifiziert ist, wird er implizit unter Verwendung der Standardqualifikationsregeln qualifiziert, die für SQL-Anweisungen gelten.

Der Eigner hat implizit alle Tabellenzugriffsrechte für die erstellte temporäre Tabelle, einschließlich der Berechtigung, die Tabelle zu löschen. Die Tabellenzugriffsrechte des Eigners können entweder einzeln oder zusammen mit der Klausel ALL erteilt (Anweisung GRANT) und entzogen (Anweisung REVOKE) werden. Eine andere Berechtigungs-ID kann auf die erstellte temporäre Tabelle nur zugreifen, wenn ihr die entsprechenden Zugriffsrechte (mit GRANT) erteilt wurden.

Indizes und SQL-Anweisungen, die Daten ändern (z. B. INSERT, UPDATE und DELETE) werden unterstützt. Indizes können nur in demselben Tabellenbereich wie die erstellte temporäre Tabelle erstellt werden.

Für die Anweisung CREATE GLOBAL TEMPORARY TABLE gilt: Sperren und Recovery sind nicht verfügbar. Die Protokollierung ist nur verfügbar, wenn die Klau-

sel LOGGED angegeben wird. Informationen zu weiteren Optionen finden Sie in der Beschreibung der Anweisung CREATE GLOBAL TEMPORARY TABLE.

Für erstellte temporäre Tabellen sind die folgenden Aktionen nicht möglich:

- Die Tabellen können keinen Sicherheitsrichtlinien zugeordnet werden.
- Die Tabellen können nicht durch Bereichspartitionierung partitioniert werden.
- Die Tabellen können nicht mit mehrdimensionalem Clustering (MDC) arbeiten.
- Die Tabellen können nicht als ITC-Tabellen genutzt werden.
- Die Tabellen können nicht als Bereichsclustertabellen (RCT) genutzt werden.
- Die Tabellen können nicht durch Replikation verteilt werden.

MQTs (Materialized Query Tables) können nicht für erstellte temporäre Tabellen erstellt werden.

Erstellte temporäre Tabellen unterstützen die folgenden Spaltentypen und Objekttypen sowie die folgenden Tabellen- oder Indexoperationen nicht:

- XML-Spalten
- Strukturierte Typen
- Verweistypen
- Integritätsbedingungen
- Indexerweiterungen
- LOAD
- LOAD TABLE
- ALTER TABLE
- RENAME TABLE
- RENAME INDEX
- REORG TABLE
- REORG INDEX
- LOCK TABLE

Weitere Informationen finden Sie in der Beschreibung der Anweisung CREATE GLOBAL TEMPORARY TABLE.

Beispiel

```
CREATE GLOBAL TEMPORARY TABLE temptbl
   LIKE empltabl
   ON COMMIT DELETE ROWS
   NOT LOGGED
   IN usr_tbsp
```

Diese Anweisung erstellt eine temporäre Tabelle mit dem Namen `temptbl`. Diese Tabelle wird mit Spalten definiert, die exakt dieselben Namen und Beschreibungen wie die Spalten der Tabelle `empltabl` haben. Die implizite Definition umfasst nur den Spaltennamen, den Datentyp, das Merkmal für die Optionalität der Dateneingabe sowie die Standardwertattribute für die Spalten in `empltabl`. Alle anderen Spaltenattribute, einschließlich der eindeutigen Integritätsbedingungen, Integritätsbedingungen über Fremdschlüssel, Trigger und Indizes, werden nicht impliziert definiert.

Eine COMMIT-Operation löscht die Zeilen immer aus der Tabelle. Wenn für die Tabelle HOLD-Cursor geöffnet sind, können sie mithilfe der Anweisung TRUNCATE gelöscht werden. Dies ist schneller, muss jedoch „normalerweise" Zeile für Zeile

geschehen. Die an der temporären Tabelle vorgenommenen Änderungen werden nicht protokolliert. Die temporäre Tabelle wird im angegebenen Tabellenbereich `usr_tbsp` für temporäre Benutzertabellen gespeichert. Dieser Tabellenbereich muss vorhanden sein, da andernfalls die Erstellung der Tabelle fehlschlägt.

Wenn eine Anwendung, die eine erstellte temporäre Tabelle instanziiert hat, die Verbindung zur Datenbank trennt, wird die für die Anwendung erstellte Instanz der erstellten temporären Tabelle gelöscht.

Erstellen von Tabellen nach vorhandenen Tabellen

Die Erstellung einer neuen Quellentabelle kann erforderlich werden, wenn bei der Ausführung der Anweisung ALTER TABLE mit der Klausel ATTACH PARTITION die Merkmale der Zieltabelle nicht ausreichend mit den Merkmalen der Quelle übereinstimmen.

Vor der Erstellung einer neuen Quellentabelle können Sie versuchen, die Abweichungen zwischen der vorhandenen Quellentabelle und der Zieltabelle zu korrigieren.

Vorbereitende Schritte

Zum Erstellen einer Tabelle müssen die Zugriffsrechte der Berechtigungs-ID der Anweisung mindestens eine der folgenden Berechtigungen und eines der folgenden Zugriffsrechte beinhalten:

- Berechtigung CREATETAB für die Datenbank und das Zugriffsrecht USE für den Tabellenbereich sowie eine der folgenden Berechtigungen (bzw. Zugriffsrechte):
  - Berechtigung IMPLICIT_SCHEMA für die Datenbank, wenn der implizite oder explizite Schemaname der Tabelle nicht existiert
  - Zugriffsrecht CREATEIN für das Schema, wenn sich der Schemaname der Tabelle auf ein vorhandenes Schema bezieht
- Berechtigung DBADM

Informationen zu diesem Vorgang

Wenn Versuche, die Abweichung zu korrigieren, fehlschlagen, wird der Fehler SQL20408N oder SQL20307N zurückgegeben.

Vorgehensweise

Gehen Sie wie folgt vor, um eine neue Quellentabelle zu erstellen:

1. Generieren Sie mit dem Befehl **db2look** eine Anweisung CREATE TABLE zur Erstellung einer Tabelle, die mit der Zieltabelle identisch ist:
   ```
   db2look -d name_der_quellendatenbank -t name_der_quellentabelle -e
   ```
2. Entfernen Sie die Partitionierungsklausel aus der Ausgabe von **db2look** und ändern Sie den Namen der erstellten Tabelle in einen neuen Namen (z. B. „sourceC").
3. Laden Sie im nächsten Schritt alle Daten aus der ursprünglichen Quellentabelle in die neu erstellte Quellentabelle 'quelleC' mit dem Befehl **LOAD FROM CURSOR**:
   ```
   DECLARE mycurs CURSOR FOR SELECT * FROM quelle
   LOAD FROM mycurs OF CURSOR REPLACE INTO quelleC
   ```

Wenn dieser Befehl fehlschlägt, weil die ursprünglichen Daten mit der Definition der Tabelle 'quelleC' nicht kompatibel sind, müssen Sie die Daten in der ursprünglichen Tabelle beim Übertragen in die Tabelle 'quelleC' umwandeln.

4. Wenn die Daten erfolgreich in 'quelleC' übertragen wurden, führen Sie die Anweisung 'ALTER TABLE *ziel* ...ATTACH *quelleC* aus.

Erstellen von Tabellen zum Zwischenspeichern von Daten

Eine *Zwischenspeichertabelle* (engl. staging table) ermöglicht eine Unterstützung von Teilaktualisierungen (inkrementelle Pflege) für MQTs (Materialized Query Table, gespeicherte Abfragetabelle), die mit REFRESH DEFERRED definiert sind. Die Zwischenspeichertabelle sammelt Änderungen, die auf die MQT angewendet werden müssen, um diese mit dem Inhalt der zugrunde liegenden Tabellen zu synchronisieren. Durch die Nutzung von Zwischenspeichertabellen wird die starke Sperrenkonkurrenz beseitigt, die durch sofort zu aktualisierende Daten verursacht wird, wenn eine unverzügliche Aktualisierung (REFRESH IMMEDIATE) der MQT angefordert ist. Außerdem brauchen MQTs nicht länger völlig neu generiert zu werden, wenn ein Befehl REFRESH TABLE ausgeführt wird.

Informationen zu diesem Vorgang

MQTs sind eine leistungsfähige Methode zum Verbessern der Antwortzeit komplexer Abfragen, insbesondere solcher Abfragen, die eventuell eine der folgenden Operationen erfordern:

- Ermitteln von Ergebnisdaten aus einer oder mehreren Dimensionen
- Joins und Ergebnisberechnung von Daten einer Gruppe von Tabellen
- Verarbeiten von Daten aus einer häufig verwendeten Teilmenge von Daten
- Erneutes Partitionieren von Daten aus einer Tabelle bzw. einem Teil einer Tabelle in einer partitionierten Datenbankumgebung

Es folgt eine Übersicht über einige der wichtigsten Einschränkungen für Zwischenspeichertabellen:

1. Die Abfrage, die zur Definition der MQT verwendet wird, für die die Zwischenspeichertabelle erstellt wird, muss inkrementell aktualisierbar sein. Das bedeutet, sie muss den gleichen Regeln wie eine MQT mit der Option zur sofortigen Aktualisierung (REFRESH IMMEDIATE) genügen.
2. Eine unterstützende Zwischenspeichertabelle ist nur für eine verzögerte Aktualisierung (REFRESH DEFERRED) möglich. Die Abfrage definiert auch die MQT, der die Zwischenspeichertabelle zugeordnet wird. Die MQT muss mit der Option REFRESH DEFERRED definiert sein.
3. Bei der Aktualisierung über die Zwischenspeichertabellen wird nur eine Aktualisierung bis zum aktuellen Zeitpunkt unterstützt.
4. Partitionierte Hierarchietabellen und partitionierte typisierte Tabellen werden nicht unterstützt. (Partitionierte Tabellen sind Tabellen, bei denen die Daten auf der Basis der in der Klausel PARTITION BY der Anweisung CREATE TABLE angegebenen Spezifikationen in mehrere Speicherobjekte untergliedert werden.)

Eine Zwischenspeichertabelle in einem inkonsistenten, unvollständigen oder schwebenden (pending) Status kann nicht zur inkrementellen Aktualisierung (Teilaktualisierung) der zugeordneten MQT verwendet werden, sofern nicht einige weitere Operationen stattfinden. Diese Operationen sorgen dafür, dass der Inhalt der Zwischenspeichertabelle mit der ihr zugeordneten MQT und deren zugrunde liegenden Tabellen konsistent wird, und nehmen die Zwischenspeichertabelle aus dem schwebenden Status heraus. Nach der Aktualisierung einer MQT wird der In-

halt der zugehörigen Zwischenspeichertabelle gelöscht und die Zwischenspeichertabelle in den Normalstatus versetzt. Eine Zwischenspeichertabelle kann mithilfe der Anweisung SET INTEGRITY und den entsprechenden Optionen auch gezielt abgeschnitten werden (PRUNE). Das Abschneiden versetzt die Zwischenspeichertabelle in einen inkonsistenten Status. Zum Beispiel erzwingt die folgende Anweisung das Abschneiden einer Zwischenspeichertabelle mit dem Namen STAGTAB1:

```
SET INTEGRITY FOR STAGTAB1 PRUNE;
```

Eine Zwischenspeichertabelle wird bei ihrer Erstellung in einen schwebenden Status versetzt und besitzt einen Anzeiger, der darauf hinweist, dass die Tabelle inkonsistent und unvollständig im Hinblick auf den Inhalt der zugrunde liegenden Tabellen und der zugeordneten MQT ist. Die Zwischenspeichertabelle muss aus dem schwebenden und inkonsistenten Status herausgenommen werden, um mit der Erfassung der Änderungen aus den zugrunde liegenden Tabellen beginnen zu können. Während des schwebenden Status schlagen alle Versuche, Änderungen an einer beliebigen der zugrunde liegenden Tabellen der Zwischenspeichertabelle vorzunehmen, ebenso fehl wie alle Versuche, die zugeordnete MQT zu aktualisieren.

Eine Zwischenspeichertabelle kann durch mehrere Methoden aus einem schwebenden Status herausgenommen werden. Zum Beispiel:

- SET INTEGRITY FOR <zwischenspeichertabelle> STAGING IMMEDIATE UNCHECKED
- SET INTEGRITY FOR <zwischenspeichertabelle> IMMEDIATE CHECKED

Unterschiede zwischen DB2-Basistabellen und temporären Tabellen

DB2-Basistabellen und die beiden Typen von temporären Tabellen weisen verschiedene Unterschiede auf.

Die folgende Tabelle enthält eine Übersicht über wichtige Unterschiede zwischen Basistabellen, erstellten temporären Tabellen und deklarierten temporären Tabellen.

Tabelle 24. Wichtige Unterschiede zwischen DB2-Basistabellen und temporären DB2-Tabellen

Unterscheidungsbereich	Unterschied
Erstellung, Persistenz und Möglichkeit der gemeinsamen Nutzung von Tabellenbeschreibungen	**Basistabellen:** Die Anweisung CREATE TABLE fügt eine Beschreibung der Tabelle in die Katalogsicht SYSCAT.TABLES ein. Die Tabellenbeschreibung ist persistent und kann über verschiedene Verbindungen hinweg gemeinsam genutzt werden. Der Name der Tabelle in der Anweisung CREATE TABLE kann qualifiziert sein. Wenn der Tabellenname nicht qualifiziert ist, wird er implizit unter Verwendung der Standardqualifikationsregeln qualifiziert, die für SQL-Anweisungen gelten.
	Erstellte temporäre Tabellen: Die Anweisung CREATE GLOBAL TEMPORARY TABLE fügt eine Beschreibung der Tabelle in die Katalogsicht SYSCAT.TABLES ein. Die Tabellenbeschreibung ist persistent und kann über verschiedene Verbindungen hinweg gemeinsam genutzt werden. Der Name der Tabelle in der Anweisung CREATE GLOBAL TEMPORARY TABLE kann qualifiziert sein. Wenn der Tabellenname nicht qualifiziert ist, wird er implizit unter Verwendung der Standardqualifikationsregeln qualifiziert, die für SQL-Anweisungen gelten.
	Deklarierte temporäre Tabellen: Die Anweisung DECLARE GLOBAL TEMPORARY TABLE fügt keine Beschreibung der Tabelle in den Katalog ein. Die Tabellenbeschreibung besteht nicht über die Dauer der Verbindung hinaus fort, die die Anweisung DECLARE GLOBAL TEMPORARY TABLE abgesetzt hat, und die Beschreibung ist nur dieser Verbindung bekannt. Daher könnte jede Verbindung potenziell eine eigene Beschreibung derselben deklarierten temporären Tabelle haben. Der Name der Tabelle in der Anweisung DECLARE GLOBAL TEMPORARY TABLE kann qualifiziert sein. Wenn der Tabellenname qualifiziert ist, muss das Schemaqualifikationsmerkmal SESSION verwendet werden. Wenn der Tabellenname nicht qualifiziert ist, wird implizit das Qualifikationsmerkmal SESSION verwendet.
Tabelleninstanziierung und Möglichkeit zur gemeinsamen Datennutzung	**Basistabellen:** Die Anweisung CREATE TABLE erstellt eine leere Instanz der Tabelle, und alle Verbindungen verwenden diese eine Instanz der Tabelle. Die Tabelle und die Daten sind persistent.
	Erstellte temporäre Tabellen: Die Anweisung CREATE GLOBAL TEMPORARY TABLE erstellt keine Instanz der Tabelle. Der erste implizite oder explizite Verweis auf die Tabelle einer OPEN-, SELECT-, INSERT-, UPDATE- oder DELETE-Operation, die von einem Programm, das die Verbindung verwendet, ausgeführt wird, erstellt eine leere Instanz der betreffenden Tabelle. Jede Verbindung, die auf die Tabelle verweist, hat eine eigene, für sie spezifische Instanz der Tabelle, und die Instanz bleibt nicht über die Dauer der Verbindung hinaus bestehen.
	Deklarierte temporäre Tabellen: Die Anweisung DECLARE GLOBAL TEMPORARY TABLE erstellt eine leere Instanz der Tabelle für die Verbindung. Jede Verbindung, die die Tabelle deklariert, hat eine eigene, für sie spezifische Instanz der Tabelle, und die Instanz bleibt nicht über die Dauer der Verbindung hinaus bestehen.

Tabelle 24. Wichtige Unterschiede zwischen DB2-Basistabellen und temporären DB2-Tabellen (Forts.)

Unterscheidungsbereich	Unterschied
Verweise auf die Tabelle während der Verbindung	**Basistabellen:** Verweise auf den Tabellennamen in mehreren Verbindungen beziehen sich auf dieselbe eine persistente Tabellenbeschreibung und auf dieselbe Instanz auf dem aktuellen Server. Wenn der Tabellenname, auf den verwiesen wird, nicht qualifiziert ist, wird er implizit unter Verwendung der Standardqualifikationsregeln qualifiziert, die für SQL-Anweisungen gelten.
	Erstellte temporäre Tabellen: Verweise auf den Tabellennamen in mehreren Verbindungen beziehen sich auf dieselbe eine persistente Tabellenbeschreibung, jedoch auf eine jeweils andere Instanz der Tabelle für jede Verbindung auf dem aktuellen Server. Wenn der Tabellenname, auf den verwiesen wird, nicht qualifiziert ist, wird er implizit unter Verwendung der Standardqualifikationsregeln qualifiziert, die für SQL-Anweisungen gelten.
	Deklarierte temporäre Tabellen: Verweise auf den Tabellennamen in mehreren Verbindungen beziehen sich auf eine spezielle Beschreibung und eine spezielle Instanz der Tabelle für jede Verbindung auf dem aktuellen Server. Verweise auf den Tabellennamen in einer SQL-Anweisung (d. h. einer anderen als der Anweisung DECLARE GLOBAL TEMPORARY TABLE) müssen das Schemaqualifikationsmerkmal SESSION enthalten. Wenn der Tabellenname nicht mit SESSION qualifiziert ist, wird der Verweis als Verweis auf eine Basistabelle interpretiert.
Zugriffsrechte und Berechtigungen für Tabellen	**Basistabellen:** Der Eigner hat implizit alle Tabellenzugriffsrechte für die Tabelle sowie die Berechtigung, die Tabelle zu löschen. Die Tabellenzugriffsrechte des Eigners können entweder einzeln oder zusammen mit der Klausel ALL erteilt (Anweisung GRANT) und entzogen (Anweisung REVOKE) werden. Eine andere Berechtigungs-ID kann auf die Tabelle nur zugreifen, wenn ihr die entsprechenden Zugriffsrechte für die Tabelle mit GRANT erteilt wurden.
	Erstellte temporäre Tabellen: Der Eigner hat implizit alle Tabellenzugriffsrechte für die Tabelle sowie die Berechtigung, die Tabelle zu löschen. Die Tabellenzugriffsrechte des Eigners können entweder einzeln oder zusammen mit der Klausel ALL erteilt (Anweisung GRANT) und entzogen (Anweisung REVOKE) werden. Eine andere Berechtigungs-ID kann auf die Tabelle nur zugreifen, wenn ihr die entsprechenden Zugriffsrechte für die Tabelle mit GRANT erteilt wurden.
	Deklarierte temporäre Tabellen: Die Gruppe PUBLIC hat implizit alle Tabellenzugriffsrechte für die Tabelle ohne die Berechtigung GRANT und hat auch die Berechtigung, die Tabelle zu löschen. Diese Tabellenzugriffsrechte können nicht erteilt oder entzogen werden. Jede Berechtigungs-ID kann auf die Tabelle zugreifen, ohne dass ihr Zugriffsrechte für die Tabelle mit GRANT erteilt werden müssten.
Indizes und Unterstützung für weitere SQL-Anweisungen	**Basistabellen:** Indizes und SQL-Anweisungen, die Daten ändern (INSERT, UPDATE, DELETE usw.) werden unterstützt. Indizes können sich in anderen Tabellenbereichen befinden.
	Erstellte temporäre Tabellen: Indizes und SQL-Anweisungen, die Daten ändern (INSERT, UPDATE, DELETE usw.) werden unterstützt. Indizes können sich nur in demselben Tabellenbereich wie die Tabelle befinden.
	Deklarierte temporäre Tabellen: Indizes und SQL-Anweisungen, die Daten ändern (INSERT, UPDATE, DELETE usw.) werden unterstützt. Indizes können sich nur in demselben Tabellenbereich wie die Tabelle befinden.

Tabelle 24. Wichtige Unterschiede zwischen DB2-Basistabellen und temporären DB2-Tabellen (Forts.)

Unterscheidungsbereich	Unterschied
Sperren, Protokollieren und Recovery	**Basistabellen:** Sperren, Protokollierung und Recovery sind verfügbar.
	Erstellte temporäre Tabellen: Sperren und Recovery sind nicht verfügbar. Die Protokollierung ist jedoch verfügbar, wenn LOGGED explizit angegeben wird. Eine rückgängig machende Recovery (Rollback von Änderungen bis zu einem Sicherungspunkt oder bis zum letzten Commitpunkt) wird nur unterstützt, wenn LOGGED *explizit* angegeben wird.
	Deklarierte temporäre Tabellen: Sperren und Recovery sind nicht verfügbar. Die Protokollierung ist nur verfügbar, wenn LOGGED explizit oder implizit angegeben wird. Eine rückgängig machende Recovery (Rollback von Änderungen bis zu einem Sicherungspunkt oder bis zum letzten Commitpunkt) wird unterstützt, wenn LOGGED explizit oder implizit angegeben wird.

Ändern von Tabellen

Dieser Abschnitt enthält Themen, die das Ändern von Tabellen erläutern.

Ändern von Tabellen

Für das Ändern von Tabellen stehen einige nützliche Optionen zur Verfügung, die zu beachten sind. Dazu gehören zum Beispiel die Option ALTER COLUMN SET DATA TYPE und die Option für unbegrenzt viele für REORG empfohlene Operationen, die innerhalb einer einzigen Transaktion ausgeführt werden können.

Unterstützung von SET DATA TYPE in der Anweisung ALTER TABLE

Die Option ALTER COLUMN SET DATA TYPE in der Anweisung ALTER TABLE unterstützt alle kompatiblen Typen.

Das Ändern des Datentyps einer Spalte kann Datenverlust zur Folge haben. Einige Arten dieses Verlusts entsprechen den Typumwandlungsregeln. Zum Beispiel können Leerzeichen aus Zeichenfolgen entfernt werden, ohne dass ein Fehler zurückgegeben wird, und die Konvertierung eines DECIMAL-Typs in einen INTEGER-Typ führt zu einem Abschneiden. Zur Vermeidung unerwarteter Fehler, wie zum Beispiel Überlauffehler, Abschneidefehler oder beliebiger anderer Arten von Fehlern, die von der Datentypumwandlung zurückgegeben werden, werden vorhandene Spaltendaten durchsucht und Nachrichten über Zeilen, die Konflikte verursachen, in das Benachrichtigungsprotokoll geschrieben. Standardwerte für Spalten werden ebenfalls überprüft, um sicherzustellen, dass sie dem neuen Datentyp entsprechen.

Wenn ein Durchsuchen der Daten zu keinen Fehlernachrichten führt, wird der Spaltentyp auf den neuen Datentyp gesetzt und die vorhandenen Spaltendaten werden in den neuen Datentyp umgesetzt. Wenn ein Fehler gemeldet wird, schlägt die Ausführung der Anweisung ALTER TABLE fehl.

Das Ändern einer VARCHAR-, VARGRAPHIC- oder LOB-Spalte in einen Datentyp, der in der Vorrangliste der Datentypen früher aufgeführt ist (siehe Abschnitt *Umstufung von Datentypen*), wird nicht unterstützt.

Beispiel

Ändern des Datentyps der Spalte SALES in der Tabelle SALES von INTEGER in SMALLINT:

```
alter table sales alter column sales set data type smallint
DB20000I  Der SQL-Befehl wurde erfolgreich ausgeführt.
```

Ändern des Datentyps der Spalte REGION in der Tabelle SALES von VARCHAR(15) in VARCHAR(14):

```
alter table sales alter column region set data type varchar(14)
...
SQL0190N  In ALTER TABLE "ADMINISTRATOR.SALES" wurden Attribute für die Spalte
"REGION" angegeben, die mit der vorhandenen Spalte nicht kompatibel sind. SQLSTATE=42837
```

Ändern eines Spaltentyps in einer Basistabelle (Dabei gibt es Sichten und Funktionen, die direkt oder indirekt von der Basistabelle abhängig sind.):

```
create table t1 (c1 int, c2 int);

create view v1 as select c1, c2 from t1;
create view v2 as select c1, c2 from v1;

create function foo1()
 language sql
 returns int
 return select c2 from t1;

create view v3 as select c2 from v2
 where c2 = foo1();

create function foo2 ()
 language sql
 returns int
 return select c2 from v3;

alter table t1
 alter column c1
  set data type smallint;

select * from v2;
```

Die Anweisung ALTER TABLE, die den Spaltentyp von INTEGER in den kleineren Typ SMALLINT umsetzt, inaktiviert die Sichten v1, v2 und v3 sowie die Funktion foo2. Unter der Semantik der verzögerten Reaktivierung werden durch `select * from v2` die Sichten v1 und v2 erfolgreich reaktiviert und die Spalten sowohl in der Sicht v1 als auch in der Sicht v2 in den Datentyp SMALLINT geändert. Die Sicht v3 und die Funktion foo2 werden jedoch nicht reaktiviert, weil auf sie nach der Inaktivierung nicht verwiesen wird und sie sich über der Sicht v2 in der Kette der Abhängigkeitshierarchie befinden. Unter der Semantik zur unverzüglichen Reaktivierung werden durch die Anweisung ALTER TABLE alle abhängigen Objekte erfolgreich reaktiviert.

Mehrere ALTER TABLE-Operationen innerhalb einer UOW

Bestimmte ALTER TABLE-Operationen, wie das Löschen einer Spalte, das Ändern eines Spaltentyps oder das Ändern der Optionalität der Dateneingabe für eine Spalte, können die Tabelle in den Status 'REORG anstehend' versetzen. In diesem Status können zahlreiche Abfragen nicht ausgeführt werden. Sie müssen eine Tabellenreorganisation ausführen, bevor die Tabelle für bestimmte Abfragetypen zur

Verfügung steht. Jedoch können Sie für die Tabelle, auch wenn sie sich im Status 'REORG anstehend' befindet, mehrere ALTER TABLE-Operationen ausführen, bevor die Reorganisation erfolgt.

Ab DB2 Version 9.7 können Sie eine unbegrenzte Anzahl von Anweisungen ALTER TABLE innerhalb einer *UOW* (Unit of Work, Arbeitseinheit) ausführen. Nach Ausführung von drei UOWs, die solche Operationen enthalten, muss jedoch ein Befehl REORG TABLE ausgeführt werden.

Ändern der Eigenschaften einer MQT (Materialized Query Table)

Mit einigen Einschränkungen können Sie eine MQT (Materialized Query Table, gespeicherte Abfragetabelle) in eine reguläre Tabelle bzw. eine reguläre Tabelle in eine MQT ändern. Bei anderen Tabellentypen als regulären Tabellen und MQTs ist eine Änderung des Tabellentyps nicht möglich. Sie können zum Beispiel keine replizierte MQT in eine reguläre Tabelle oder umgekehrt ändern.

Informationen zu diesem Vorgang

Nach dem Ändern einer regulären Tabelle in eine MQT wird die Tabelle in den Status 'Festlegen der Integrität anstehend' versetzt. Bei einer derartigen Änderung muss der `Fullselect` in der Definition der MQT mit der ursprünglichen Tabellendefinition übereinstimmen. Dies bedeutet Folgendes:

- Die Anzahl der Spalten muss gleich sein.
- Die Spaltennamen und -positionen müssen übereinstimmen.
- Die Datentypen müssen identisch sein.

Wenn die MQT für eine Originaltabelle definiert wird, kann diese Originaltabelle nicht selbst in eine MQT geändert werden. Wenn für die Originaltabelle Trigger, Prüfungen auf Integritätsbedingungen, referenzielle Integritätsbedingungen oder definierte eindeutige Indizes definiert sind, kann diese nicht in eine MQT geändert werden. Wenn Sie die Tabelleneigenschaften ändern, um eine MQT zu definieren, ist es nicht zulässig, die Tabelle innerhalb derselben Anweisung ALTER TABLE in irgendeiner Form zu ändern.

Beim Ändern einer regulären Tabelle in eine MQT kann der `Fullselect` der Definition der MQT nicht direkt oder indirekt über Sichten, Aliasnamen oder andere MQTs auf die Originaltabelle verweisen.

Geben Sie den folgenden Befehl ein, um eine MQT in eine reguläre Tabelle zu ändern:

```
ALTER TABLE sumtable
   SET SUMMARY AS DEFINITION ONLY
```

Geben Sie den folgenden Befehl ein, um eine reguläre Tabelle in eine MQT zu ändern:

```
ALTER TABLE regtable
  SET SUMMARY AS <fullselect>
```

Die Einschränkungen, die beim Ändern einer regulären Tabelle in eine MQT für den `Fullselect` gelten, stimmen größtenteils mit den Einschränkungen überein, die für das Erstellen einer Übersichtstabelle mit der Anweisung CREATE SUMMARY TABLE gelten.

Aktualisieren der Daten in einer MQT (Materialized Query Table)

Sie können die Daten in einer oder mehreren MQTs (Materialized Query Table, gespeicherte Abfragetabelle) mithilfe der Anweisung REFRESH TABLE aktualisieren. Die Anweisung kann in ein Anwendungsprogramm eingebettet oder dynamisch abgesetzt werden. Zur Verwendung dieser Anweisung ist die Berechtigung DATAACCESS bzw. das Zugriffsrecht CONTROL für die zu aktualisierende Tabelle erforderlich.

Beispiel

Das folgende Beispiel zeigt, wie Sie die Daten in einer MQT aktualisieren können:

```
REFRESH TABLE SUMTAB1
```

Ändern von Spalteneigenschaften

Mithilfe der Anweisung ALTER TABLE können Sie Spalteneigenschaften wie die Optionalität der Dateneingabe, LOB-Optionen, den Bereich, Attribute für Integritätsbedingungen und Komprimierung, Datentypen usw. ändern.

Die vollständigen Informationen finden Sie in der Beschreibung der Anweisung ALTER TABLE.

Vorbereitende Schritte

Zum Ändern einer Tabelle müssen Sie mindestens über eines der folgenden Zugriffsrechte (bzw. Berechtigungen) für die zu ändernde Tabelle verfügen:

- Zugriffsrecht ALTER
- Zugriffsrecht CONTROL
- Berechtigung DBADM
- Zugriffsrecht ALTERIN für das Schema der Tabelle

Sie müssen über die Berechtigung DBADM verfügen, um Folgendes auszuführen:

- Ändern der Definition einer vorhandenen Spalte.
- Bearbeiten und Testen des SQL beim Ändern von Tabellenspalten.
- Überprüfen zusammengehöriger Objekte beim Ändern von Tabellenspalten.

Vorgehensweise

Gehen Sie wie folgt vor, um Spalteneigenschaften zu ändern:

Führen Sie die Anweisung ALTER TABLE aus. Geben Sie über die Befehlszeile zum Beispiel Folgendes ein:

```
ALTER TABLE EMPLOYEE
   ALTER COLUMN WORKDEPT
      SET DEFAULT '123'
```

Hinzufügen und Löschen von Spalten

Mithilfe der Anweisung ALTER TABLE können Sie vorhandenen Tabellen Spalten hinzufügen, indem Sie die Klausel ADD COLUMN verwenden, und Spalten aus vorhandenen Tabellen entfernen, indem Sie die Klausel DROP COLUMN verwenden. Die Tabelle darf keine typisierte Tabelle sein.

Informationen zu diesem Vorgang

In allen vorhandenen Zeilen der Tabelle wird die neue Spalte auf ihren Standardwert gesetzt. Die neue Spalte ist die letzte Spalte der Tabelle. Das heißt, wenn zuvor *n* Spalten vorhanden waren, ist die hinzugefügte Spalte die Spalte *n*+1. Durch das Hinzufügen der neuen Spalte darf die Gesamtbytezahl aller Spalten die Zeilengrößenbegrenzung nicht überschreiten.

Vorgehensweise

- Setzen Sie zum Hinzufügen einer Spalte die Anweisung ALTER TABLE mit der Klausel ADD COLUMN ab. Beispiel:

```
ALTER TABLE SALES
  ADD COLUMN SOLD_QTY
  SMALLINT NOT NULL DEFAULT 0
```

- Setzen Sie zum Löschen einer Spalte die Anweisung ALTER TABLE mit der Klausel DROP COLUMN ab. Beispiel:

```
ALTER TABLE SALES
  DROP COLUMN SOLD_QTY
```

Ändern von Spaltendefinitionen mit der Klausel DEFAULT

Die Klausel DEFAULT stellt einen Standardwert für eine Spalte für den Fall bereit, dass bei einer Einfügung (INSERT) kein Wert angegeben wird, oder wird als DEFAULT in INSERT- oder UPDATE-Anweisungen angegeben. Wenn nach dem Schlüsselwort DEFAULT kein bestimmter Wert angegeben wird, hängt der Standardwert vom Datentyp ab. Ist eine Spalte mit dem Datentyp XML oder mit einem strukturierten Typ definiert, kann die Klausel DEFAULT nicht angegeben werden.

Informationen zu diesem Vorgang

Wenn die Klausel DEFAULT in einer Spaltendefinition nicht angegeben wird, wird der Nullwert als Standardwert für die Spalte verwendet (siehe „Standardspalten- und Datentypdefinitionen” auf Seite 329).

Bestimmte Typen von Werten, die mit dem Schlüsselwort DEFAULT angegeben werden können, finden Sie in den Informationen zur Anweisung ALTER TABLE.

Modifizieren der Eigenschaft GENERATED oder IDENTITY einer Spalte

Die Eigenschaft GENERATED oder IDENTITY einer Spalte in einer Tabelle kann mithilfe der Klausel ALTER COLUMN der Anweisung ALTER TABLE hinzugefügt oder gelöscht werden.

Sie können eine der folgenden Aktionen ausführen:

- Wenn Sie mit einer vorhandenen, nicht generierten Spalte (ohne die Eigenschaft GENERATED) arbeiten, können Sie ein Attribut mit einem GENERATED-Ausdruck hinzufügen. Die modifizierte Spalte wird dann zu einer generierten Spalte.
- Wenn Sie mit einer vorhandenen generierten Spalte (mit der Eigenschaft GENERATED) arbeiten, können Sie das Attribut mit dem GENERATED-Ausdruck löschen. Die modifizierte Spalte wird dann zu einer normalen, nicht generierten Spalte.
- Wenn Sie mit einer vorhandenen Nicht-Identitätsspalte (ohne die Eigenschaft IDENTITY) arbeiten, können Sie ein IDENTITY-Attribut hinzufügen. Die modifizierte Spalte wird dann zu einer Identitätsspalte.

- Wenn Sie mit einer vorhandenen Identitätsspalte arbeiten, können Sie das IDENTITY-Attribut löschen. Die modifizierte Spalte wird dann zu einer normalen, nicht generierten Nicht-Identitätsspalte.
- Wenn Sie mit einer vorhandenen Identitätsspalte arbeiten, können Sie das Attribut der Spalte von GENERATED ALWAYS in GENERATED BY DEFAULT ändern. Der umgekehrte Fall ist ebenfalls möglich. Das heißt, Sie können das Attribut der Spalte von GENERATED BY DEFAULT in GENERATED ALWAYS ändern. Dies ist nur bei der Arbeit mit einer Identitätsspalte möglich.
- Sie können das DEFAULT-Attribut aus einer benutzerdefinierten Standardspalte löschen. Wenn Sie dies tun, ist der neue Standardwert null.
- Sie können innerhalb derselben Anweisung ALTER COLUMN das vorhandene DEFAULT-, IDENTITY- oder GENERATED-Attribut löschen und anschließend ein neues DEFAULT-, IDENTITY- oder GENERATED-Attribut definieren.
- Für die Anweisungen CREATE TABLE und ALTER TABLE ist das Schlüsselwort „ALWAYS" in der Klausel für die generierte Spalte optional. Das heißt, in diesem Fall sind GENERATED ALWAYS und GENERATED äquivalent.

Ändern von Spaltendefinitionen

Mit der Anweisung ALTER TABLE können Sie Spalten löschen oder Typen und Attribute von Spalten ändern. Sie können zum Beispiel den Wert für die Länge einer vorhandenen VARCHAR- oder VARGRAPHIC-Spalte erhöhen. Die Anzahl der Zeichen kann bis zu einem Wert erhöht werden, der von der verwendeten Seitengröße abhängig ist.

Informationen zu diesem Vorgang

Wenn Sie zum Ändern des Standardwerts einer Spalte den neuen Standardwert definiert haben, wird der neue Wert bei allen nachfolgenden SQL-Operationen verwendet, wenn die Verwendung des Standardwerts für die Spalte angegeben ist. Der neue Wert muss den Regeln für die Zuordnung entsprechen und unterliegt ebenfalls den Einschränkungen, die unter der Anweisung CREATE TABLE dokumentiert sind.

Anmerkung: Der Standardwert generierter Spalten kann mithilfe dieser Anweisung nicht geändert werden.

Wenn Sie diese Tabellenattribute mithilfe von SQL ändern, ist es nicht mehr erforderlich, die Tabelle zu löschen und erneut zu stellen, da dies ein zeitaufwendiger und komplexer Prozess sein kann, wenn Objektabhängigkeiten vorhanden sind.

Vorgehensweise

- Geben Sie in die Befehlszeile Folgendes ein, um die Länge und den Typ einer Spalte einer vorhandenen Tabelle zu ändern:

```
   ALTER TABLE tabellenname
ALTER COLUMN spaltenname
modifikationstyp
```

  Mit der folgenden Anweisung können Sie zum Beispiel die Zeichenzahl einer Spalte auf 4.000 Zeichen erhöhen:

```
ALTER TABLE t1
   ALTER COLUMN spalte1
   SET DATA TYPE VARCHAR(4000)
```

  Das folgende Beispiel zeigt, wie Sie eine Anweisung wie die folgende verwenden, um einer Spalte einen neuen VARGRAPHIC-Wert zuzuordnen:

```
ALTER TABLE t1
   ALTER COLUMN spalte2
   SET DATA TYPE VARGRAPHIC(2000)
```

Sie können die Spalte einer typisierten Tabelle nicht ändern. Sie können einer vorhandenen Verweistypspalte ohne definierten Bereich jedoch einen Bereich hinzufügen. Beispiel:

```
ALTER TABLE t1
   ALTER COLUMN spaltet1
   ADD SCOPE typtab1
```

- Geben Sie den folgenden Befehl ein, um eine Spalte so zu ändern, dass LOB-Daten inline gespeichert werden können:

```
   ALTER TABLE tabellenname
ALTER COLUMN spaltenname
SET INLINE LENGTH neue_LOB-länge
```

  Wenn Sie zum Beispiel möchten, dass LOB-Daten mit bis zu 1000 Byte in einer Basistabellenzeile gespeichert werden, verwenden Sie eine Anweisung ähnlich der folgenden:

```
ALTER TABLE t1
   ALTER COLUMN spalte1
   SET INLINE LENGTH 1004
```

  In diesem Fall wird die Länge nicht auf 1000, sondern auf 1004 gesetzt. Dies geschieht deswegen, weil für inline gespeicherte LOB-Daten zusätzliche 4 Byte für Speicher über die Größe der eigentlichen LOB-Daten hinaus erforderlich sind.

- Geben Sie in die Befehlszeile Folgendes ein, um den Standardwert einer Spalte einer vorhandenen Tabelle zu ändern:

```
   ALTER TABLE tabellenname
ALTER COLUMN spaltenname
SET DEFAULT 'neuer_standardwert'
```

  Mit einer Anweisung wie der folgenden können Sie zum Beispiel den Standardwert für eine Spalte ändern:

```
ALTER TABLE t1
   ALTER COLUMN spalte1
   SET DEFAULT '123'
```

Umbenennen von Tabellen und Spalten

Mithilfe der Anweisung RENAME können Sie eine vorhandene Tabelle umbenennen. Zum Umbenennen von Spalten verwenden Sie die Anweisung ALTER TABLE.

Informationen zu diesem Vorgang

Wenn Tabellen umbenannt werden, dürfen keine vorhandenen Definitionen (von Sichten oder MQTs), Trigger, SQL-Funktionen oder Integritätsbedingungen auf die Quellentabelle verweisen. Darüber hinaus darf die Tabelle keine generierten Spalten (außer Identitätsspalten) enthalten, und sie darf keine übergeordnete oder abhängige Tabelle sein. Katalogeinträge werden mit dem neuen Tabellennamen aktualisiert. Weitere Informationen und Beispiele finden Sie in der Beschreibung der Anweisung RENAME.

Die Klausel RENAME COLUMN ist eine Option in der Anweisung ALTER TABLE. Sie können eine vorhandene Spalte in einer Basistabelle in einen neuen Namen umbenennen, ohne dass gespeicherte Daten verloren gehen oder Zugriffsrechte bzw. Richtlinien für die kennsatzbasierte Zugriffssteuerung (LBAC), die der Tabelle zugeordnet sind, beeinträchtigt werden.

Nur die Umbenennung von Basistabellenspalten wird unterstützt. Die Umbenennung von Spalten in Sichten, MQTs (Materialized Query Tables), deklarierten und erstellten temporären Tabellen und anderen tabellenähnlichen Objekten wird nicht unterstützt.

Die Semantik für die Inaktivierung und Reaktivierung im Zusammenhang mit Umbenennungsoperationen für Spalten ist derjenigen ähnlich, die für Löschoperationen für Spalten gilt: Alle abhängigen Objekte werden inaktiviert (ungültig gemacht). Die Reaktivierung aller abhängigen Objekte nach einer Umbenennungsoperation für Spalten erfolgt immer unverzüglich nach der Inaktivierung, auch wenn der Datenbankkonfigurationsparameter **auto_reval** auf den Wert DISABLED gesetzt ist.

Das folgende Beispiel zeigt die Umbenennung einer Spalte mit der Anweisung ALTER TABLE:

```
ALTER TABLE org RENAME COLUMN deptnumb TO deptnum
```

Informationen zum Ändern der Definition vorhandener Spalten finden Sie im Abschnitt "Ändern von Spalteneigenschaften" oder in der Beschreibung der Anweisung ALTER TABLE.

Recovery funktionsunfähiger Übersichtstabellen

Übersichtstabellen können infolge eines widerrufenen Zugriffsrechts SELECT für eine zugrunde liegende Tabelle *funktionsunfähig* werden.

Informationen zu diesem Vorgang

Die folgenden Schritte können Sie bei der Recovery einer funktionsunfähigen Übersichtstabelle unterstützen:

- Stellen Sie fest, mit welcher Anweisung die Übersichtstabelle zu Anfang erstellt wurde. Diese Information können Sie der Spalte TEXT der Katalogsicht SYSCAT.VIEW entnehmen.
- Erstellen Sie die Übersichtstabelle erneut, indem Sie die Anweisung CREATE SUMMARY TABLE mit demselben Übersichtstabellennamen und derselben Definition verwenden.
- Verwenden Sie die Anweisung GRANT, um alle Zugriffsrechte, die zuvor für die Übersichtstabelle erteilt waren, erneut zu erteilen. (Beachten Sie, dass alle für eine funktionsunfähig gewordene Übersichtstabelle erteilten Zugriffsrechte widerrufen werden.)

Wenn Sie eine funktionsunfähige Übersichtstabelle nicht wiederherstellen möchten, können Sie sie explizit mit der Anweisung DROP TABLE löschen, oder Sie können eine neue Übersichtstabelle mit demselben Namen, aber einer anderen Definition erstellen.

Eine funktionsunfähige Übersichtstabelle hat nur noch Einträge in den Katalogsichten SYSCAT.TABLES und SYSCAT.VIEWS. Alle Einträge in den Katalogsichten SYSCAT.TABDEP, SYSCAT.TABAUTH, SYSCAT.COLUMNS und SYSCAT.COLAUTH werden entfernt.

Anzeigen von Tabellendefinitionen

Mithilfe der Katalogsichten SYSCAT.TABLES und SYSCAT.COLUMNS können Tabellendefinitionen angezeigt werden. In der Katalogsicht SYSCAT.COLUMNS stellt jede Zeile eine Spalte dar, die für eine Tabelle, eine Sicht oder einen Kurznamen definiert ist. Zum Anzeigen der Daten in den Spalten können Sie die Anweisung SELECT verwenden.

Informationen zu diesem Vorgang

Sie können auch die folgenden Sichten und Tabellenfunktionen zum Anzeigen von Tabellendefinitionen verwenden:

- ADMINTEMPCOLUMNS (Verwaltungssicht)
- ADMINTEMPTABLES (Verwaltungssicht)
- ADMIN_GET_TEMP_COLUMNS (Tabellenfunktion) - Spalteninformationen für temporäre Tabellen abrufen
- ADMIN_GET_TEMP_TABLES (Tabellenfunktion) - Informationen für temporäre Tabellen abrufen

Löschen von Tabellen

Eine Tabelle kann mit einer Anweisung DROP TABLE gelöscht werden.

Informationen zu diesem Vorgang

Beim Löschen einer Tabelle wird die Zeile in der Systemkatalogsicht SYSCAT.TABLES gelöscht, die die Informationen über die Tabelle enthält, während alle anderen von der Tabelle abhängigen Objekte auf verschiedene Weise davon betroffen sind. Zum Beispiel:

- Alle Spaltennamen werden gelöscht.
- Die für Spalten der Tabelle erstellten Indizes werden gelöscht.
- Alle Sichten, die auf der Tabelle basieren, werden als funktionsunfähig markiert.
- Alle Zugriffsrechte auf die gelöschte Tabelle und abhängige Sichten werden implizit widerrufen.
- Alle referenziellen Integritätsbedingungen, in denen die Tabelle eine übergeordnete oder abhängige Tabelle ist, werden gelöscht.
- Alle Pakete und im Cache zwischengespeicherten dynamischen SQL- und XQuery-Anweisungen, die von der gelöschten Tabelle abhängig sind, werden als ungültig markiert und bleiben ungültig, bis die abhängigen Objekte neu erstellt werden. Dazu gehören Pakete, die von einer zu löschenden übergeordneten Tabelle über der untergeordneten Tabelle in der Hierarchie abhängen.
- Verweisspalten, für die die gelöschte Tabelle als Bereich des Verweises definiert ist, haben keinen Bereich mehr.
- Die Definition eines Aliasnamens für die Tabelle wird nicht berührt, da ein Aliasname auch ohne Tabelle vorhanden sein kann.
- Alle von der gelöschten Tabelle abhängigen Trigger werden als funktionsunfähig markiert.

Einschränkungen

Eine einzelne Tabelle kann nicht gelöscht werden, wenn sie eine untergeordnete Tabelle hat.

Vorgehensweise

- Verwenden Sie zum Löschen einer Tabelle die Anweisung DROP TABLE.

  Mit der folgenden Anweisung wird die Tabelle DEPARTMENT gelöscht:

  ```
  DROP TABLE DEPARTMENT
  ```

- Verwenden Sie zum Löschen aller Tabellen in einer Tabellenhierarchie die Anweisung DROP TABLE HIERARCHY.

  In der Anweisung DROP TABLE HIERARCHY muss die Stammtabelle der zu löschenden Hierarchie angegeben werden. Zum Beispiel:

  ```
  DROP TABLE HIERARCHY person
  ```

Ergebnisse

Zwischen dem Löschen einer Tabellenhierarchie und dem Löschen einer bestimmten Tabelle bestehen folgende Unterschiede:

- DROP TABLE HIERARCHY aktiviert keine Löschtrigger, wie sie von einzelnen Anweisungen DROP TABLE aktiviert würden. Beim Löschen einer einzelnen untergeordneten Tabelle würden zum Beispiel Löschtrigger für die dazugehörigen übergeordneten Tabellen aktiviert.
- DROP TABLE HIERARCHY erstellt keine Protokolleinträge für die einzelnen Zeilen der gelöschten Tabellen. Das Löschen der Hierarchie wird stattdessen als ein einziges Ereignis protokolliert.

Löschen von MQTs oder Zwischenspeichertabellen

Sie können eine MQT (Materialized Query Table, gespeicherte Abfragetabelle) oder Zwischenspeichertabelle nicht ändern, sondern nur löschen. Alle Indizes, Primärschlüssel, Fremdschlüssel und Prüfungen auf Integritätsbedingungen, die auf die Tabelle verweisen, werden gelöscht. Alle Sichten und Trigger, die auf die Tabelle verweisen, werden funktionsunfähig gemacht. Alle Pakete, die von einem gelöschten oder als funktionsunfähig markierten Objekt abhängen, werden inaktiviert (ungültig gemacht).

Informationen zu diesem Vorgang

Eine MQT kann explizit mit der Anweisung DROP TABLE oder implizit durch das Löschen einer der zugrunde liegenden Tabellen gelöscht werden.

Eine Zwischenspeichertabelle kann explizit mit der Anweisung DROP TABLE oder implizit durch das Löschen ihrer zugeordneten MQT gelöscht werden.

Vorgehensweise

Geben Sie in die Befehlszeile die folgende Anweisung ein, um eine MQT oder Zwischenspeichertabelle zu löschen:

```
DROP TABLE tabellenname
```

Mit der folgenden Anweisung wird die MQT XT gelöscht:

```
DROP TABLE XT
```

Time Travel Query (TTQ) über temporale Tabellen

Sie können temporale Tabellen verwenden, um Ihren Daten zeitbasierte Statusinformationen zuzuordnen. Die Daten in Tabellen ohne temporale Unterstützung sind aktuell (d. h. zum gegenwärtigen Zeitpunkt) gültig. Im Unterschied dazu können Daten in temporalen Tabellen für einen Zeitraum gültig sein, der vom Datenbanksystem und/oder von Benutzeranwendungen definiert wird.

In vielen Geschäftssituationen ist die Speicherung und Verwaltung zeitbasierter Daten erforderlich. Datenbanken ohne diese Funktionalität machen aufwändige und komplexe Prozesse erforderlich, um eine Infrastruktur für die Unterstützung zeitbezogener Daten bereitzustellen. In Datenbanken mit temporalen Tabellen können zeitbasierte Daten ohne zusätzliche Anwendungslogik gespeichert und abgerufen werden. Beispielsweise kann in einer Datenbank das Verlaufsprotokoll für eine Tabelle gespeichert sein (d. h. die Inhalte gelöschter Zeilen oder die ursprünglichen Werte für Zeilen, die aktualisiert wurden), damit Sie frühere Stände Ihrer Daten abrufen können. Außerdem können Sie einer Datenzeile einen Datumsbereich zuordnen, um festzulegen, wann die Daten von Ihren Anwendungen oder Geschäftsregeln als gültig angesehen werden.

Eine temporale Tabelle erfasst den Zeitraum, in dem eine Datenzeile gültig ist. Dieser Zeitraum wird in der temporalen Tabelle durch zwei Spalten für Datum/Zeit definiert. Ein Zeitraum verfügt über eine Beginnspalte und eine Endspalte. Die Beginnspalte gibt den Anfang des Zeitraums an und die Endspalte das Ende des Zeitraums. Der Beginnwert für einen Zeitraum wird inklusiv angegeben, der Endwert jedoch exklusiv. Beispiel: Eine Datenzeile mit der Zeitraumangabe 1. Januar bis 1. Februar ist gültig vom 1. Januar bis zum 31. Januar um Mitternacht.

Die beiden folgenden Zeitraumtypen werden unterstützt:

Systemzeitraum
Ein Systemzeitraum besteht aus zwei Spalten mit vom Datenbankmanager verwalteten Werten, die den Zeitraum angeben, in dem eine Zeile aktuell ist. Die Beginnspalte enthält einen Zeitmarkenwert für den Erstellungszeitpunkt der Zeile. Die Endspalte enthält einen Zeitmarkenwert für den Zeitpunkt, an dem die Zeile aktualisiert oder gelöscht wurde. Wenn eine temporale Tabelle für Systemzeitraum erstellt wird, enthält sie die aktuell aktiven Zeilen. Jeder temporalen Tabelle für Systemzeitraum ist eine Protokolltabelle zugeordnet, die alle geänderten Zeilen enthält.

Anwendungszeitraum
Ein Anwendungszeitraum besteht aus zwei Spalten mit vom Benutzer oder von Anwendungen angegebenen Werten, die den Gültigkeitszeitraum für eine Zeile angeben. Die Beginnspalte gibt an, ab welchem Zeitpunkt eine Zeile gültig ist. Die Endspalte gibt an, ab welchem Zeitpunkt eine Zeile nicht mehr gültig ist. Eine Tabelle mit einem Anwendungszeitraum wird als temporale Tabelle für Anwendungszeitraum bezeichnet.

Durch Abfragen der Systemkatalogsicht SYSCAT.TABLES können Sie prüfen, ob eine Tabelle temporale Unterstützung bietet. Beispiel:

```
SELECT TABSCHEMA, TABNAME, TEMPORALTYPE FROM SYSCAT.TABLES
```

Die Rückgabewerte für `TEMPORALTYPE` sind wie folgt definiert:

A Temporale Tabelle für Anwendungszeitraum

B Bitemporale Tabelle

N Nichttemporale Tabelle

S Temporale Tabelle für Systemzeitraum

Temporale Tabellen für Systemzeitraum

Eine temporale Tabelle für Systemzeitraum verwaltet archivierte Versionen der zugehörigen Tabellenzeilen. In einer temporalen Tabelle für Systemzeitraum können Sie aktuelle Versionen Ihrer Daten speichern. Die zugehörige Protokolltabelle ermöglicht das transparente Speichern Ihrer aktualisierten und gelöschten Datenzeilen.

Eine temporale Tabelle für Systemzeitraum enthält einen Zeitraum für SYSTEM_TIME mit Spalten, die aufzeichnen, wann die Aktualität der Daten in einer Zeile beginnt bzw. endet. Außerdem verwendet der Datenbankmanager den Zeitraum für SYSTEM_TIME, um archivierte Versionen der einzelnen Tabellenzeilen aufzuzeichnen, wenn Aktualisierungen oder Löschvorgänge ausgeführt werden. Der Datenbankmanager speichert diese Zeilen in einer Protokolltabelle, die ausschließlich einer temporalen Tabelle für Systemzeitraum zugeordnet ist. Durch das Hinzufügen der Versionierung wird die temporale Tabelle für Systemzeitraum mit der Protokolltabelle verknüpft. Über eine temporale Tabelle für Systemzeitraum können Ihre Abfragen auf die aktuell gültigen Daten zugreifen sowie Daten für bestimmte Zeitpunkte in der Vergangenheit abrufen.

Eine temporale Tabelle für Systemzeitraum enthält auch eine Spalte für die Transaktionsstart-ID. Diese Spalte erfasst den Zeitpunkt, an dem die Ausführung einer Transaktion gestartet wurde, die sich auf die Zeile auswirkt. Wenn innerhalb einer einzigen SQL-Transaktion mehrere Zeilen eingefügt oder aktualisiert werden, sind die Werte der Spalten für die Transaktionsstart-ID für alle Zeilen identisch und innerhalb der Werte, die von anderen Transaktionen für diese Spalte generiert werden, eindeutig. Über diesen gemeinsamen Spaltenwert für die Start-ID können Sie alle Zeilen in der Tabelle identifizieren, die von derselben Transaktion geschrieben wurden.

Protokolltabellen

Für jede temporale Tabelle für Systemzeitraum ist eine Protokolltabelle erforderlich. Beim Aktualisieren oder Löschen einer Zeile in einer temporalen Tabelle für Systemzeitraum fügt der Datenbankmanager eine Kopie der alten Zeile in die zugehörige Protokolltabelle ein. Dieses Speichern der alten Daten der temporalen Tabelle für Systemzeitraum bietet Ihnen die Möglichkeit, Daten für bestimmte Zeitpunkte in der Vergangenheit abzurufen.

Das Speichern von Zeilendaten ist nur möglich, wenn Namen, Reihenfolge und Datentypen der Spalten in der temporalen Tabelle für Systemzeitraum und in der Protokolltabelle identisch sind. Mit der Klausel LIKE in der Anweisung CREATE TABLE können Sie eine Protokolltabelle mit denselben Namen und Beschreibungen wie in der temporalen Tabelle für Systemzeitraum erstellen. Beispiel:

```
CREATE TABLE
employees_history LIKE employees IN hist_space;
```

Eine vorhandene Tabelle kann als Protokolltabelle verwendet werden, sofern die in der Beschreibung der Klausel USE HISTORY für die Anweisung ALTER TABLE angegebenen Einschränkungen berücksichtigt werden.

Nach dem Erstellen einer Protokolltabelle fügen Sie eine Versionierung hinzu, um die temporale Tabelle für Systemzeitraum mit der Protokolltabelle zu verknüpfen.

```
ALTER TABLE employees ADD VERSIONING USE HISTORY TABLE employees_history;
```

Bei aktivierter Versionssteuerung unterliegt eine Protokolltabelle folgenden Regeln und Einschränkungen:

- Eine Protokolltabelle kann nicht explizit gelöscht werden. Sie kann nur implizit gelöscht werden, wenn die zugehörige temporale Tabelle für Systemzeitraum gelöscht wird.
- Protokolltabellenspalten können nicht explizit hinzugefügt, gelöscht oder geändert werden.
- Eine Protokolltabelle darf nicht als übergeordnetes, untergeordnetes oder auf sich selbst verweisendes Element in einer referenziellen Integritätsbedingung definiert werden. Der Zugriff auf die Protokolltabelle wird eingeschränkt, um kaskadierende Aktionen in der Protokolltabelle zu vermeiden.
- Ein Tabellenbereich, der eine Protokolltabelle ohne die zugehörige temporale Tabelle für Systemzeitraum enthält, kann nicht gelöscht werden.

Eine Protokolltabelle sollte nur in Ausnahmefällen explizit geändert werden. Eine solche Änderung kann die Möglichkeit zum Prüfen der Datenprotokollierung einer temporalen Tabelle für Systemzeitraum gefährden. Der Zugriff auf eine Protokolltabelle sollte eingeschränkt werden, um die darin enthaltenen Daten zu schützen.

Bei normalem Betrieb werden für eine Protokolltabelle Einfüge- und Leseaktivitäten am häufigsten ausgeführt. Aktualisierungen und Löschvorgänge sind selten. Das Fehlen von Aktualisierungen und Löschvorgängen bedeutet, dass Protokolltabellen in der Regel keinen freien Speicherplatz aufweisen, der zum Einfügen neuer Zeilen verwendet werden könnte. Wenn Zeileneinfügungen in die Protokolltabelle negative Auswirkungen auf die Auslastungsleistung haben, können Sie die Suche nach freiem Speicherplatz vermeiden, indem Sie die Definition der Protokolltabelle mit der Option APPEND ON ändern. Bei dieser Option entfällt der zusätzliche Verarbeitungsaufwand, der durch die Suche nach freiem Speicherplatz entsteht. Sie bewirkt, dass neue Zeilen an das Ende der Tabelle angefügt werden.

```
ALTER TABLE employees_history APPEND ON;
```

Beim Löschen einer temporalen Tabelle für Systemzeitraum werden die zugehörige Protokolltabelle und alle für die Protokolltabelle definierten Indizes implizit gelöscht. Um den Verlust von Protokolldaten beim Löschen einer temporalen Tabelle für Systemzeitraum zu vermeiden, können Sie die Protokolltabelle mit dem Attribut RESTRICT ON DROP erstellen oder das Attribut RESTRICT ON DROP zu der erstellten Protokolltabelle hinzufügen.

```
CREATE TABLE employees_history LIKE employees WITH RESTRICT ON DROP;
```

Da für Protokolltabellen mehr Einfügungen als Löschvorgänge ausgeführt werden, nimmt der Umfang Ihrer Protokolltabellen stetig zu und der benötigte Speicherbedarf wächst. Es kann eine komplexe Aufgabe sein, zur Bereinigung Ihrer Protokolltabellen zu entscheiden, welche Zeilen nicht mehr benötigt werden. Sie müssen den Wert Ihrer einzelnen Datensätze verstehen. Manche Teile des Inhalts, wie z. B. Kundenverträge, sind möglicherweise tabu und können niemals entfernt werden. Andere Datensätze dagegen, wie Informationen zu Websitebesuchern, können ohne Bedenken bereinigt werden. Häufig ist nicht das Alter einer Zeile der ausschlaggebende Faktor dafür, dass sie bereinigt und archiviert werden kann, sondern eine bestimmte Geschäftslogik. Die folgende Liste enthält einige mögliche Regeln für die Bereinigung:

- Bereinigen Sie Zeilen, die von einer vom Benutzer ausgewählten Abfrage ausgewählt wurden, die Geschäftsregeln widerspiegelt.
- Bereinigen Sie Zeilen, die ein bestimmtes Alter überschreiten.

- Bereinigen Sie Protokollzeilen, wenn mehr als 'n' Versionen für diesen Datensatz vorhanden sind (behalten Sie nur die neuesten 'n' Versionen bei).
- Bereinigen Sie Protokollzeilen, wenn der Datensatz aus der zugeordneten temporalen Tabelle für Systemzeitraum gelöscht wurde (wenn keine aktuellen Versionen vorhanden sind).

Es gibt mehrere Möglichkeiten, um alte Daten in regelmäßigen Abständen aus einer Protokolltabelle zu bereinigen:

- Verwenden Sie die Bereichspartitionierung und heben Sie die Zuordnung alter Partitionen zur Protokolltabelle auf.
- Verwenden Sie DELETE-Anweisungen, um Zeilen aus der Tabelle zu entfernen. Wenn Sie DELETE-Anweisungen verwenden, sollten Sie die folgenden Richtlinien beachten:
  - Reorganisieren Sie die Protokolltabelle in regelmäßigen Abständen, um den freien Speicherplatz, der nach Löschoperationen entsteht, freizugeben.
  - Stellen Sie sicher, dass die Protokolltabelle nicht zum Verwenden der Option APPEND ON geändert wurde, und so Einfügungen für die Suche nach freiem Speicherplatz zulässig sind.

Zeitraum für SYSTEM_TIME

Die Zeitraumspalten für SYSTEM_TIME in einer temporalen Tabelle für Systemzeitraum geben an, wann die Version einer Zeile aktuell ist.

Der Zeitraum für SYSTEM_TIME enthält zwei TIMESTAMP(12)-Spalten, deren Werte vom Datenbankmanager generiert werden. Diese Spalten müssen als NOT NULL mit einer gültigen Option GENERATED ALWAYS AS definiert werden. Die Anfangsspalte des Zeitraums muss eine Beginnspalte für Zeile und die Endspalte des Zeitraum muss eine Endspalte für Zeile sein.

```
CREATE TABLE policy_info
(
 policy_id       CHAR(4) NOT NULL,
 coverage        INT NOT NULL,
 sys_start   TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW BEGIN,
 sys_end  TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW END,
 ts_id   TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS TRANSACTION START ID,
 PERIOD SYSTEM_TIME (sys_start, sys_end)
) IN policy_space;
```

Beginnspalte für Zeile

Diese Spalte gibt an, wann die Zeilendaten aktuell wurden. Der Datenbankmanager generiert den Wert für diese Spalte anhand einer Leseabfrage der Systemuhr im Moment der Ausführung der ersten Datenänderungsanweisung innerhalb der Transaktion, von der die Zeile generiert wird. Wenn innerhalb einer einzigen SQL-Transaktion mehrere Zeilen eingefügt oder aktualisiert werden, sind die Werte der Beginnspalten für Zeile in allen betroffenen Zeilen identisch. Die Werte dieser Beginnspalten für Zeile sind eindeutig im Vergleich zu den Werten für die Beginnspalten für Zeile der anderen Transaktionen. Eine Beginnspalte für Zeile ist als Anfangsspalte für einen SYSTEM_TIME-Zeitraum erforderlich, der für jede temporale Tabelle für Systemzeitraum definiert werden muss.

Wenn eine vorhandene reguläre Tabelle geändert wird, um eine temporale Tabelle für Systemzeitraum zu erzeugen, wird eine Beginnspalte für Zeile zu der Tabelle hinzugefügt. Die Beginnspalte für Zeile wird mit dem Standardwert `0001-01-01-00.00.00.000000000000` für den Datentypwert TIMESTAMP(12) aller vorhandenen Zeilen gefüllt.

Endspalte für Zeile

Diese Spalte gibt an, ab wann die Zeilendaten nicht mehr aktuell waren. Für die Zeilen einer Protokolltabelle gibt die Endspalte für Zeile an, wann die Zeile zur Protokolltabelle hinzugefügt wurde. Die Zeilen in der temporalen Tabelle für Systemzeitraum sind per Definition aktuell, d. h. die Endspalte für Zeile wird mit einem Standardwert für den Datentyp TIMESTAMP(12) gefüllt (z. B. `9999-12-30-00.00.00.000000000000`). Eine Endspalte für Zeile ist als Endspalte für einen SYSTEM_TIME-Zeitraum erforderlich, der für jede temporale Tabelle für Systemzeitraum definiert werden muss.

Wenn eine vorhandene reguläre Tabelle geändert wird, um eine temporale Tabelle für Systemzeitraum zu erzeugen, wird eine Endspalte für Zeile zu der Tabelle hinzugefügt. Die Endspalte für Zeile aller vorhandenen Zeilen wird mit dem maximalen Wert für den Datentyp TIMESTAMP(12) gefüllt (Standardwert: `9999-12-30-00.00.00.000000000000`).

Da es sich bei Beginn- und Endspalte für Zeile um generierte Spalten handelt, wird keine implizite Prüfung auf Integritätsbedingung für SYSTEM_TIME generiert, die gewährleistet, dass der Wert für eine Endspalte in einer temporalen Tabelle für Systemzeitraum größer ist als der Wert für die zugehörige Beginnspalte. Im Gegensatz dazu wird bei einer temporalen Tabelle für Anwendungszeitraum eine Prüfung auf Integritätsbedingung für BUSINESS_TIME generiert. Eine Zeile, deren Wert für die Endspalte kleiner ist als der Wert für die zugehörige Beginnspalte, kann nicht zurückgegeben werden, wenn beim Abfragen der Tabelle eine Zeitraumangabe verwendet wird. Sie können eine Integritätsbedingung definieren, um sicherzustellen, dass der Wert für die Endspalte größer ist als der Wert für die Beginnspalte. Diese Garantie ist hilfreich für die Unterstützung von Operationen, die explizit Daten in diese generierten Spalten eingeben (z. B. Ladeoperationen).

Der Datenbankkonfigurationsparameter **`systime_period_adj`** wird verwendet, um anzugeben, welche Aktion ausgeführt werden soll, wenn eine Protokollzeile für eine temporale Tabelle für Systemzeitraum mit einem Endspaltenwert generiert wird, der kleiner ist als der Wert der Beginnspalte.

Erstellen einer temporalen Tabelle für Systemzeitraum

Die Erstellung einer temporalen Tabelle für Systemzeitraum resultiert in einer Tabelle, die Datenänderungen protokolliert und Langzeitversionen dieser Daten archiviert.

Informationen zu diesem Vorgang

Fügen Sie bei der Erstellung einer temporalen Tabelle für Systemzeitraum Attribute ein, die angeben, ob die Daten in einer Zeile aktuell sind und Transaktionen Auswirkungen auf die Daten haben:

- Fügen Sie Zeilenbeginn und -endspalten ein, mit deren Hilfe der Zeitraum von SYSTEM_TIME feststellt, ob eine Zeile aktuell ist.
- Fügen Sie eine Spalte für die Transaktionsstart-ID ein, die die Startzeiten von Transaktionen erfasst, die Auswirkungen auf Zeilen haben.
- Erstellen Sie eine Protokolltabelle, in die alte Zeilen aus der temporalen Tabelle für Systemzeitraum eingefügt werden.
- Fügen Sie die Versionssteuerung hinzu (ADD VERSIONING), um eine Verbindung zwischen der temporalen Tabelle für Systemzeitraum und der Protokolltabelle herzustellen.

Die Spalten für Zeilenbeginn, Zeilenende und die Transaktionsstart-ID können als implizit verdeckt (IMPLICITLY HIDDEN) definiert werden. Da diese Spalten und die zugehörigen Einträge vom Datenbankmanager generiert werden, kann das Verdecken dieser Spalten potenziell negative Auswirkungen auf die Anwendungen minimieren. Verdeckte Spalten sind nur dann verfügbar, wenn auf sie verwiesen wird. Beispiel:

- Eine für eine Tabelle ausgeführte Abfrage SELECT * gibt keine implizit verdeckten Spalten in der Ergebnistabelle zurück.
- INSERT-Anweisungen erwarten keine Werte für implizit verdeckte Spalten.
- Die Befehle LOAD, IMPORT und EXPORT können den Änderungswert *includeimplicitlyhidden* für die Arbeit mit implizit verdeckten Spalten verwenden.

Eine temporale Tabelle für Systemzeitraum kann in einer referenziellen Integritätsbedingung als übergeordnete oder untergeordnete Tabelle definiert werden. Referenzielle Integritätsbedingungen werden jedoch nur auf die aktuellen Daten angewendet (d. h. die Daten in der temporalen Tabelle für Systemzeitraum). Die Integritätsbedingungen werden nicht auf die zugehörige Protokolltabelle angewendet. Zur Minimierung von Inkonsistenzen für den Fall, dass eine temporale Tabelle für Systemzeitraum eine untergeordnete Tabelle in einer referenziellen Integritätsbedingung ist, sollte die übergeordnete Tabelle ebenfalls eine temporale Tabelle für Systemzeitraum sein.

Anmerkung: Die generierten Spalten für Zeilenbeginn, Zeilenende und Transaktionsstart-ID sind bei der Erstellung einer temporalen Tabelle für Systemzeitraum erforderlich; mit diesen generierten Spalten kann jedoch auch eine reguläre Tabelle erstellt werden.

Das Beispiel im folgenden Abschnitt zeigt die Erstellung einer Tabelle, in der die Daten von Versicherungspolicen für die Kunden einer Versicherungsgesellschaft gespeichert werden.

Vorgehensweise

Gehen Sie wie folgt vor, um eine temporale Tabelle für Systemzeitraum zu erstellen:

1. Erstellen Sie eine Tabelle mit dem Attribut SYSTEM_TIME. Beispiel:

```
CREATE TABLE policy_info
(
 policy_id       CHAR(4) NOT NULL,
 coverage        INT NOT NULL,
 sys_start   TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW BEGIN,
 sys_end  TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW END,
 ts_id   TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS TRANSACTION START ID,
 PERIOD SYSTEM_TIME (sys_start, sys_end)
) IN policy_space;
```

2. Erstellen Sie eine Protokolltabelle. Beispiel:

```
CREATE TABLE hist_policy_info
(
 policy_id       CHAR(4) NOT NULL,
 coverage        INT NOT NULL,
 sys_start  TIMESTAMP(12) NOT NULL,
 sys_end  TIMESTAMP(12) NOT NULL,
 ts_id    TIMESTAMP(12) NOT NULL
) IN hist_space;
```

Sie können auch eine Protokolltabelle mit denselben Namen und Beschreibungen wie für die Spalten der temporalen Tabelle für Systemzeitraum erstellen. Verwenden Sie dazu die Klausel LIKE der Anweisung CREATE TABLE. Beispiel:

```
CREATE TABLE hist_policy_info LIKE policy_info IN hist_space;
```

3. Fügen Sie die Versionssteuerung zur temporalen Tabelle für Systemzeitraum hinzu, um eine Verbindung zur Protokolltabelle herzustellen. Beispiel:

```
ALTER TABLE policy_info ADD VERSIONING USE HISTORY TABLE hist_policy_info;
```

Ergebnisse

In der Tabelle `policy_info` wird der Versicherungsdeckungswert für einen Kunden gespeichert. Die auf den Zeitraum SYSTEM_TIME bezogenen Spalten (`sys_start` und `sys_end`) zeigen an, ob eine Zeile für den Deckungswert aktuell ist. Die Spalte `ts_id` gibt die Startzeit der Ausführung einer Transaktion an, die Auswirkungen auf die Zeile hat.

Tabelle 25. Erstellte temporale Tabelle für Systemzeitraum (policy_info)

policy_id	coverage	sys_start	sys_end	ts_id

Die Protokolltabelle `hist_policy_info` erhält die alten Zeilen aus der Tabelle `policy_info`.

Tabelle 26. Erstellte Protokolltabelle (hist_policy_info)

policy_id	coverage	sys_start	sys_end	ts_id

Beispiel

Der folgende Abschnitt enthält weitere Beispiele zur Erstellung von temporalen Tabellen für Systemzeitraum.

Verdecken von Spalten

Das folgende Beispiel erstellt die Tabelle `policy_info`, wobei die TIMESTAMP(12)-Spalten (`sys_start`, `sys_end` und `ts_id`) als implizit verdeckt markiert sind.

```
CREATE TABLE policy_info
(
 policy_id       CHAR(4) NOT NULL,
 coverage        INT NOT NULL,
 sys_start   TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW BEGIN IMPLICITLY HIDDEN,
 sys_end  TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW END IMPLICITLY HIDDEN,
 ts_id   TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS TRANSACTION START ID IMPLICITLY HIDDEN,
 PERIOD SYSTEM_TIME (sys_start, sys_end)
) in policy_space;
```

Wenn die Protokolltabelle `hist_policy_info` mit der Klausel LIKE der Anweisung CREATE TABLE erstellt wird, übernimmt die Protokolltabelle das Attribut IMPLICITLY HIDDEN von der Tabelle `policy_info`.Wenn Sie die Klausel LIKE beim Erstellen der Protokolltabelle nicht verwenden, müssen alle Spalten, die in der temporalen Tabelle für Systemzeitraum als verdeckt markiert waren, auch in der zugehörigen Protokolltabelle als verdeckt markiert werden.

Ändern einer vorhandenen Tabelle in eine temporale Tabelle für Systemzeitraum Das folgende Beispiel fügt Zeitmarkenspalten und einen SYSTEM_TIME-

Zeitraum zu einer vorhandenen Tabelle (employees) hinzu und aktiviert die Tabellenfunktionen der temporalen Tabelle für Systemzeitraum.

```
ALTER TABLE employees
   ADD COLUMN sys_start TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW BEGIN;
ALTER TABLE employees
   ADD COLUMN sys_end TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW END;
ALTER TABLE employees
   ADD COLUMN ts_id TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS TRANSACTION START ID;
ALTER TABLE employees ADD PERIOD SYSTEM_TIME(sys_start, sys_end);
```

Sie können diese neuen Spalten verdecken, indem Sie in der Anweisung ALTER TABLE die Klausel IMPLICITLY HIDDEN angeben.

Zum Abschließen dieser Task muss eine Protokolltabelle erstellt und die Versionssteuerung hinzugefügt werden.

Einfügen von Daten in eine temporale Tabelle für Systemzeitraum

Für einen Benutzer ist das Einfügen von Daten in eine temporale Tabelle für Systemzeitraum mit dem Einfügen von Daten in eine reguläre Tabelle vergleichbar.

Informationen zu diesem Vorgang

Beim Einfügen von Daten in eine temporale Tabelle für Systemzeitraum generiert der Datenbankmanager die Werte der Zeitmarkenspalten für Zeilenbeginn und -ende. Ferner generiert der Datenbankmanager den Wert für die Transaktionsstart-ID, der die Transaktion eindeutig identifiziert, von der die Zeile eingefügt wird.

Vorgehensweise

Zum Einfügen von Daten in eine temporale Tabelle für Systemzeitraum wird die INSERT-Anweisung verwendet. Beispiel: Die folgenden Daten wurden am 31. Januar 2010 (2010-01-31) in die Tabelle eingefügt, die im Beispiel unter „ Erstellen einer temporalen Tabelle für Systemzeitraum erstellt wurde."

```
INSERT INTO policy_info(policy_id, coverage)
   VALUES('A123',12000);

INSERT INTO policy_info(policy_id, coverage)
   VALUES('B345',18000);

INSERT INTO policy_info(policy_id, coverage)
   VALUES('C567',20000);
```

Ergebnisse

Die Tabelle policy_info enthält nun die folgenden Versicherungsdeckungsdaten. Die Einträge der Spalten sys_start, sys_end und ts_id wurden vom Datenbankmanager generiert.

Tabelle 27. Einer temporalen Tabelle für Systemzeitraum hinzugefügte Daten (policy_info)

policy_id	coverage	sys_start	sys_end	ts_id
A123	12000	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
B345	18000	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
C567	20000	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000

Die Protokolltabelle `his_policy_info` bleibt leer, weil bei einer Einfügung keine Protokollzeilen generiert werden.

Tabelle 28. Protokolltabelle (hist_policy_info) nach Einfügung

policy_id	coverage	sys_start	sys_end	ts_id

Anmerkung: Die Beginnspalte für Zeile `sys_start` gibt den Zeitpunkt an, an dem die Zeilendaten aktuell wurden. Der Datenbankmanager generiert diesen Wert anhand einer Leseabfrage der Systemuhr im Moment der Ausführung der ersten Datenänderungsanweisung innerhalb der Transaktion, von der die Zeile generiert wird. Der Datenbankmanager generiert auch die Spalte für die Transaktionsstart-ID `ts_id`, mit der der Zeitpunkt erfasst wird, an dem die Ausführung für eine Transaktion gestartet wurde, die Auswirkungen auf die Zeile hat. In vielen Fällen sind die Zeitmarkenwerte für diese beiden Spalten gleich, da sie das Ergebnis der Ausführung derselben Transaktion sind.

Wenn mehrere Transaktionen dieselbe Zeile aktualisieren, können Konflikte mit der Zeitmarke auftreten. Der Datenbankmanager kann diese Konflikte lösen, indem Anpassungen bei den Zeitmarkenwerten der Beginnspalten für Zeile vorgenommen werden. In diesen Fällen würden sich die Werte in der Beginnspalte für Zeile und der Spalte für Transaktions-ID unterscheiden. Der Abschnitt **Beispiel** bietet in „ Aktualisieren einer temporalen Tabelle für Systemzeitraum" weitere Details zu Zeitmarkenanpassungen.

Aktualisieren von Daten in einer temporalen Tabelle für Systemzeitraum

Das Aktualisieren von Daten in einer temporalen Tabelle für Systemzeitraum führt dazu, dass Zeilen in die zugehörige Protokolltabelle eingefügt werden.

Informationen zu diesem Vorgang

Zusätzlich zur Aktualisierung von Werten angegebener Spalten in den Zeilen der temporalen Tabelle für Systemzeitraum fügt die Anweisung UPDATE eine Kopie der vorhandenen Zeile in die zugehörige Protokolltabelle ein. Die Protokollzeile wird im Rahmen der Transaktion generiert, die auch die Zeile aktualisiert.Wenn von einzelnen Transaktionen mehrere Aktualisierungen an derselben Zeile vorgenommen werden, wird nur eine Protokollzeile generiert und diese Zeile gibt den Status des Datensatzes wieder, bevor Änderungen von der Transaktion ausgeführt wurden.

Anmerkung: Wenn mehrere Transaktionen dieselbe Zeile aktualisieren, können Zeitmarkenkonflikte auftreten. Wenn diese Konflikte auftreten, legt die Einstellung für den Datenbankkonfigurationsparameter „systime_period_adj" fest, ob Zeitmarkenanpassungen vorgenommen werden oder ob Transaktionen fehlschlagen sollen. Das Beispiel **Mehrere Änderungen an einer Zeile durch unterschiedliche Transaktionen** im Abschnitt **Weitere Beispiele** bietet weitere Details. Anwendungsprogrammierer sollten die Verwendung von SQLCODE- oder SQLSTATE-Werten erwägen, um potenzielle auf die Anpassung von Zeitmarkenwerten bezogene Rückkehrcodes über SQL-Anweisungen zu handhaben.

Vorgehensweise

Zum Aktualisieren von Daten in einer temporalen Tabelle für Systemzeitraum wird die Anweisung UPDATE verwendet. Beispiel: Es wurde festgestellt, dass die Versi-

cherungsdeckungswerte für einen Kunden fehlerhaft sind. Die folgenden Daten wurden am 28. Februar 2011 (2011-02-28) in der Beispieltabelle aktualisiert, in die Daten eingefügt wurden (siehe „Einfügen von Daten in eine temporale Tabelle für Systemzeitraum").
Die folgende Tabelle enthält die ursprünglichen Daten der Tabelle `policy_info`.

Tabelle 29. Ursprüngliche Daten in der temporalen Tabelle für Systemzeitraum (policy_info)

policy_id	coverage	sys_start	sys_end	ts_id
A123	12000	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
B345	18000	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
C567	20000	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000

- Die Deckung (coverage) für die Police C567 soll 25000 betragen.

```
UPDATE policy_info
   SET coverage = 25000
   WHERE policy_id = 'C567';
```

  Die Aktualisierung der Police C567 hat Auswirkungen auf die temporale Tabelle für Systemzeitraum sowie die zugehörige Protokolltabelle und führt dazu, dass die folgenden Aktionen ausgeführt werden:

  1. Der Deckungswert für die Zeile mit der Police C567 wird auf 25000 aktualisiert.
  2. In der temporalen Tabelle für Systemzeitraum aktualisiert der Datenbankmanager die Werte von `sys_start` und `ts_id` auf das Datum der Aktualisierung.
  3. Die ursprüngliche Zeile wird in die Protokolltabelle versetzt. Der Datenbankmanager aktualisiert den Wert für `sys_end` auf das Datum der Aktualisierung. Diese Zeile kann als gültige Deckung für die Police C567 von 2010-01-31-22.31.33.495925000000 bis 2011-02-28-09.10.12.649592000000 interpretiert werden.

Tabelle 30. Aktualisierte Daten in der temporalen Tabelle für Systemzeitraum (policy_info)

policy_id	coverage	sys_start	sys_end	ts_id
A123	12000	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
B345	18000	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
C567	25000	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000

Tabelle 31. Protokolltabelle (hist_policy_info) nach Aktualisierung

policy_id	coverage	sys_start	sys_end	ts_id
C567	20000	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.649592000000	2010-01-31-22.31.33.495925000000

Weitere Beispiele

Der folgende Abschnitt enthält weitere Beispiele zur Aktualisierung von temporalen Tabellen für Systemzeitraum.

Zeitangaben

Im folgenden Beispiel wird im Rahmen der Tabellenaktualisierung ein Zeitraum angegeben. Die folgende Aktualisierung wird nach der im Abschnitt **Vorgehensweise** beschriebenen Aktualisierung ausgeführt.

```
UPDATE (SELECT * FROM policy_info
   FOR SYSTEM_TIME AS OF '2010-01-31-22.31.33.495925000000')
   SET coverage = coverage + 1000;
```

Diese Aktualisierung gibt einen Fehler zurück, da implizit versucht wird, Protokollzeilen zu aktualisieren. SELECT führt explizit eine Abfrage für die Tabelle `policy_info` und implizit eine Abfrage für die zugehörige Protokolltabelle (`hist_policy_info`) aus. Normalerweise würde die Anweisung SELECT die Zeile C567 in `hist_policy_info` zurückgeben, aber Zeilen in einer Protokolltabelle, auf die implizit zugegriffen wurde, können nicht aktualisiert werden.

Mehrere Änderungen an einer Zeile durch unterschiedliche Transaktionen

Im folgenden Beispiel führen zwei Transaktionen gleichzeitig SQL-Anweisungen für die Tabelle `policy_info` aus. In diesem Beispiel werden die Zeitmarken der Einfachheit halber durch einen Platzhalter und nicht durch einen Systemuhrwert angegeben. Beispielsweise wird anstelle von 2010-01-31-22.31.33.495925000000 der Platzhalter T1 verwendet. Platzhalter mit einer höheren Nummerierung weisen auf nachfolgende Aktionen innerhalb der Transaktion hin. T5 ist zum Beispiel ein späterer Zeitpunkt als T4. Beim Einfügen oder Aktualisieren mehrerer Zeilen innerhalb einer einzelnen SQL-Transaktion sind die Werte für die Zeilenbeginnspalten für alle betroffenen Zeilen identisch. Dieser Wert stammt aus dem Ablesen der Systemuhr zu dem Zeitpunkt, an dem die erste Datenänderungsanweisung in der Transaktion ausgeführt wird. Zum Beispiel verfügen alle der Transaktion ABC zugeordneten Zeiten über den Zeitwert T1.

Transaktion ABC

```
T1: INSERT INTO policy_info(policy_id, coverage)
      VALUES ('S777',7000);
```

Transaktion XYZ

```
T2: INSERT INTO policy_info
      (policy_id, coverage)
      VALUES ('T888',8000);

T3: COMMIT;
```

Transaktion ABC

```
T4: UPDATE policy_info
      SET policy_id = 'X999'
      WHERE policy_id = 'T888';

T5: INSERT INTO policy_info(policy_id, coverage)
      VALUES ('Y555',9000);

T6: COMMIT;
```

Nach den Einfügungen zu den Zeiten **T1** und **T2** sieht die Tabelle `policy_info` wie folgt aus; die Protokolltabelle `hist_policy_info`) ist leer. Der Wert **max** in der Spalte `sys_end` wird mit dem maximalen Standardwert für den Datentyp TIMESTAMP(12) gefüllt.

Tabelle 32. Einfügungen unterschiedlicher Transaktionen in die Tabelle 'policy_info'

policy_id	coverage	sys_start	sys_end	ts_id
S777	7000	T1	max	T1

Tabelle 32. Einfügungen unterschiedlicher Transaktionen in die Tabelle 'policy_info' (Forts.)

policy_id	coverage	sys_start	sys_end	ts_id
T888	8000	T2	max	T2

Nach der Aktualisierung durch die Transaktion ABC zum Zeitpunkt T4 entsprechen die Policeninformationen den Daten in den folgenden Tabellen. Alle Zeilen in der Tabelle `policy_info` geben die Einfügungs- und Aktualisierungsaktivitäten von Transaktion ABC wieder. Die Spalten `sys_start` und `ts_id` für diese Zeilen werden mit dem Zeitwert von T1 gefüllt, der den Zeitpunkt der ersten Datenänderungsanweisung in der Transaktion ABC angibt. Die von der Transaktion XYZ eingefügten Policeninformationen wurden aktualisiert und die ursprüngliche Zeile wird in die Protokolltabelle versetzt.

Tabelle 33. Unterschiedliche Transaktionen nach Aktualisierung der Tabelle 'policy_info'

policy_id	coverage	sys_start	sys_end	ts_id
S777	7000	T1	max	T1
X999	8000	T1	max	T1

Tabelle 34. Protokolltabelle nach Aktualisierung durch unterschiedliche Transaktionen

policy_id	coverage	sys_start	sys_end	ts_id
T888	8000	T2	T1	T2

Die Protokolltabelle enthält einen Wert für `sys_end`, der vor dem Wert von `sys_start` liegt. In dieser Situation könnte die Aktualisierung zum Zeitpunkt T4 nicht ausgeführt werden und die Transaktion ABC würde fehlschlagen (SQLSTATE 57062, SQLCODE SQL20528N). Zum Vermeiden solcher Fehler kann der Datenbankkonfigurationsparameter **`systime_period_adj`** auf den Wert *YES* gesetzt werden, wodurch es dem Datenbankmanager möglich ist, die Zeitmarke der Beginnspalte für Zeile anzupassen (SQLSTATE 01695, SQLCODE SQL5191W). Die Zeitmarke `sys_start` für die Aktualisierung zum Zeitpunkt T4 in Transaktion ABC wird auf den Zeitpunkt T2 zuzüglich eines Deltas (T2+delta) gesetzt. Die Anpassung bezieht sich nur auf die Aktualisierung zum Zeitpunkt T4; alle anderen von der Transaktion ABC durchgeführten Änderungen verwenden weiterhin die Zeitmarke für den Zeitpunkt T1 (z. B. die Einfügung der Police mit `policy_id` Y555). Nach dieser Anpassung und der Ausführung der Transaktion ABC enthalten die Tabellen für Versicherungspolicen die folgenden Daten:

Tabelle 35. Unterschiedliche Transaktionen nach Zeitanpassung (policy_info)

policy_id	coverage	sys_start	sys_end	ts_id
S777	7000	T1	max	T1
X999	8000	T2+delta	max	T1
Y555	9000	T1	max	T1

Tabelle 36. Protokolltabelle nach Zeitanpassung (hist_policy_info)

policy_id	coverage	sys_start	sys_end	ts_id
T888	8000	T2	T2+delta	T2

Mehrere Änderungen an einer Zeile innerhalb einer Transaktion

Im folgenden Beispiel nimmt eine Transaktion mehrere Änderungen an einer Zeile vor: Unter Verwendung der Tabellen für Versicherungspolicen

aus dem vorherigen Beispiel fährt die Transaktion ABS fort und aktualisiert die Police mit policy_id X999 zum Zeitpunkt T6 (T6 war ursprünglich eine Anweisung COMMIT).

Transaktion ABC

```
T6: UPDATE policy_info SET policy_id = 'R111' WHERE policy_id = 'X999';
T7: COMMIT;
```

Diese Zeile wurde nun wie folgt geändert:

1. Erstellt als Richtlinie T888 von Transaktion XYZ zum Zeitpunkt T2.
2. Aktualisiert als Richtlinie X999 von Transaktion ABC zum Zeitpunkt T4.
3. Aktualisiert als Richtlinie R111 von Transaktion ABC zum Zeitpunkt T6.

Wenn eine Transaktion mehrere Aktualisierungen für dieselbe Zeile vornimmt, generiert der Datenbankmanager nur für die erste Änderung eine Protokollzeile. Dies führt als Ergebnis zu den folgenden Tabellen:

Tabelle 37. Dieselbe Transaktion nach Aktualisierung (policy_info)

policy_id	coverage	sys_start	sys_end	ts_id
S777	7000	T1	max	T1
R111	8000	T1	max	T1
Y555	9000	T1	max	T1

Tabelle 38. Protokolltabelle nach Aktualisierung durch dieselbe Transaktion (hist_policy_info)

policy_id	coverage	sys_start	sys_end	ts_id
T888	8000	T2	T2+delta	T2

Der Datenbankmanager verwendet die Transaktionsstart-ID (ts_id), um die für die Zeilenänderung verantwortliche Transaktion eindeutig zu identifizieren. Beim Einfügen oder Aktualisieren mehrerer Zeilen innerhalb einer einzelnen SQL-Transaktion sind die Werte der Spalte für die Transaktionsstart-ID für alle Zeilen identisch und unterscheiden sich von den Werten, die von anderen Transaktionen für diese Spalte generiert wurden. Vor der Generierung einer Protokollzeile stellt der Datenbankmanager fest, dass sich die letzte Aktualisierung der Zeile auf die Transaktion bezieht, die zum Zeitpunkt T1 (ts_id ist T1) gestartet wurde; da dieser Wert der Transaktionsstartzeit für die Transaktion entspricht, die die aktuelle Änderung durchführt, wird keine Protokollzeile generiert. Der sys_start-Wert für die Zeile in der Tabelle policy_info wird in den Zeitpunkt T1 geändert.

Aktualisieren einer Sicht

Eine Sicht, die auf eine temporale Tabelle für Systemzeitraum oder eine bitemporale Tabelle verweist, kann nur aktualisiert werden, wenn die Sichtdefinition nicht die Klausel FOR SYSTEM_TIME enthält. Die folgende Anweisung UPDATE aktualisiert die Tabelle policy_info und generiert Protokollzeilen.

```
CREATE VIEW viewA AS SELECT * FROM policy_info;
UPDATE viewA SET col2 = col2 + 1000;
```

Eine Sicht, die auf eine temporale Tabelle für Systemzeitraum oder eine bitemporale Tabelle mit einer Sichtdefinition verweist, die die Klausel FOR SYSTEM_TIME enthält, kann durch Definieren eines Triggers INSTEAD OF aktualisierbar gemacht werden. Das folgende Beispiel aktualisiert die Tabelle regular_table.

```
CREATE VIEW viewB AS SELECT * FROM policy_info;
   FOR SYSTEM_TIME BETWEEN
   TIMESTAMP '2010-01-01 10:00:00' AND TIMESTAMP '2011-01-01 10:00:00';

CREATE TRIGGER update INSTEAD OF UPDATE ON viewB
   REFERENCING NEW AS n FOR EACH ROW
   UPDATE regular_table SET col1 = n.id;

UPDATE viewB set id = 500;
```

Löschen von Daten aus einer temporalen Tabelle für Systemzeitraum

Das Löschen von Daten aus einer temporalen Tabelle für Systemzeitraum entfernt Zeilen aus der Tabelle und fügt Zeilen zur zugeordneten Protokolltabelle hinzu. Die Zeilen werden mit den entsprechenden Systemzeitmarken hinzugefügt.

Informationen zu diesem Vorgang

Neben dem Löschen der angegebenen Zeilen aus der temporalen Tabelle für Systemzeitraum versetzt die Anweisung DELETE FROM eine Kopie der vorhandenen Zeile in die zugehörige Protokolltabelle, bevor die Zeile aus der temporalen Tabelle für Systemzeitraum gelöscht wird.

Vorgehensweise

Zum Löschen von Daten aus einer temporalen Tabelle für Systemzeitraum wird die Anweisung DELETE FROM verwendet. Beispiel: Der Inhaber der Police B345 entscheidet sich für die Aufhebung der Versicherungsdeckung. Die Daten wurden am 1. September 2011 (2011-09-01) aus der Tabelle gelöscht, die im Abschnitt „ Aktualisieren von Daten in einer bitemporalen Tabelle” aktualisiert wurde.

```
DELETE FROM policy_info WHERE policy_id = 'B345';
```

Ergebnisse

Die ursprünglichen Daten aus der Tabelle `policy_info` lauten wie folgt:

Tabelle 39. Daten in der temporalen Tabelle für Systemzeitraum (policy_info) vor der DELETE-Anweisung

policy_id	coverage	sys_start	sys_end	ts_id
A123	12000	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
B345	18000	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
C567	25000	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000

Das Löschen der Police B345 hat Auswirkungen auf die temporale Tabelle für Systemzeitraum sowie die zugehörige Protokolltabelle und führt dazu, dass die folgenden Aktionen ausgeführt werden:

1. Die Zeile, für die der Wert der Spalte `policy_id` B345 lautet, wird aus der temporalen Tabelle für Systemzeitraum gelöscht.
2. Die ursprüngliche Zeile wird in die Protokolltabelle versetzt. Der Datenbankmanager aktualisiert den Wert der Spalte `sys_end` auf das Löschdatum.

Tabelle 40. Daten in der temporalen Tabelle für Systemzeitraum (policy_info) nach der DELETE-Anweisung

policy_id	coverage	sys_start	sys_end	ts_id
A123	12000	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
C567	25000	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000

Tabelle 41. Protokolltabelle (hist_policy_info) nach dem Löschvorgang

policy_id	coverage	sys_start	sys_end	ts_id
C567	20000	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.649592000000	2010-01-31-22.31.33.495925000000
B345	18000	2010-01-31-22.31.33.495925000000	2011-09-01-12.18.22.959254000000	2010-01-31-22.31.33.495925000000

Beispiel

Der folgende Abschnitt enthält weitere Beispiele zu Löschoperationen für temporale Tabellen für Systemzeitraum.

Zeitangaben

Im folgenden Beispiel wird im Rahmen der DELETE-Anweisung ein Zeitraum angegeben. Der folgende Löschvorgang wird nach dem im Abschnitt **Vorgehensweise** beschriebenen Löschvorgang ausgeführt.

```
DELETE FROM (SELECT * FROM policy_info
   FOR SYSTEM_TIME AS OF '2010-01-31-22.31.33.495925000000')
   WHERE policy_id = C567;
```

Diese DELETE-Anweisung gibt einen Fehler zurück. Die Anweisung SELECT führt explizit eine Abfrage für die Tabelle `policy_info` und implizit eine Abfrage für die zugehörige Protokolltabelle (`hist_policy_info`) aus. Normalerweise würde die Anweisung SELECT die Zeile mit einem Spaltenwert von C567 für `policy_id` in der Tabelle `hist_policy_info` zurückgeben, aber die Zeilen in einer Protokolltabelle, auf die implizit zugegriffen wurde, können nicht gelöscht werden.

Abfragen von Daten in einer temporalen Tabelle für Systemzeitraum

Durch das Abfragen einer temporalen Tabelle für Systemzeitraum können Ergebnisse für einen bestimmten Zeitpunkt oder Zeitraum zurückgegeben werden. Diese Ergebnisse können aktuelle Werte und früher protokollierte Werte enthalten.

Informationen zu diesem Vorgang

Beim Abfragen einer temporalen Tabelle für Systemzeitraum kann FOR SYSTEM_TIME in die FROM-Klausel eingefügt werden. Mithilfe von FOR SYSTEM_TIME-Spezifikationen kann der aktuelle und der frühere Status Ihrer Daten abgefragt werden. Die Zeiträume werden wie folgt angegeben:

AS OF *wert1*

Enthält alle Zeilen, bei denen der Anfangswert für den Zeitraum kleinergleich *wert1* und der Endwert für den Zeitraum größer als *wert1* ist. Dies ermöglicht Ihnen, Ihre Daten ab einem bestimmten Zeitpunkt abzufragen.

FROM *wert1* **TO** *wert2*

Enthält alle Zeilen, bei denen der Anfangswert für den Zeitraum größer-

gleich *wert1* und der Endwert für den Zeitraum kleiner als *wert2* ist. Dies bedeutet, dass die Anfangszeit des Zeitraums eingeschlossen ist, die Endzeit jedoch nicht.

BETWEEN *wert1* **AND** *wert2*
Enthält alle Zeilen, bei denen sich ein beliebiger Zeitraum mit einem beliebigen Zeitpunkt zwischen *wert1* und *wert2* überschneidet. Eine Zeile wird zurückgegeben, wenn der Anfangswert für den Zeitraum kleiner-gleich *wert2* und der Endwert für den Zeitraum größer als *wert1* ist.

Im folgenden Abschnitt sind einige Beispielabfragen aufgelistet.

Vorgehensweise

Zum Abfragen einer temporalen Tabelle für Systemzeitraum wird die Anweisung SELECT verwendet. Beispielsweise fordert jede der folgenden Abfragen Daten zu Versicherungspolicen aus den im Abschnitt Löschen von Daten aus einer temporalen Tabelle für Systemzeitraum aufgeführten Ergebnistabellen an. Jede Abfrage verwendet eine Variante der Spezifikation FOR SYSTEM_TIME.
Die Tabelle `policy_info` und die zugehörige Protokolltabelle sehen wie folgt aus:

Tabelle 42. Temporale Tabelle für Systemzeitraum: policy_info

policy_id	coverage	sys_start	sys_end	ts_id
A123	12000	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
C567	25000	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000

Tabelle 43. Protokolltabelle: hist_policy_info

policy_id	coverage	sys_start	sys_end	ts_id
C567	20000	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.649592000000	2010-01-31-22.31.33.495925000000
B345	18000	2010-01-31-22.31.33.495925000000	2011-09-01-12.18.22.959254000000	2010-01-31-22.31.33.495925000000

- Abfrage ohne Zeitraumangabe. Beispiel:

```
SELECT policy_id, coverage
   FROM policy_info
   where policy_id = 'C567'
```

  Diese Abfrage gibt eine Zeile zurück. Von der Anweisung SELECT wird nur die Tabelle `policy_info` abgefragt. Die Protokolltabelle wird nicht abgefragt, da FOR SYSTEM_TIME nicht angegeben wurde.

```
C567, 25000
```

- Abfrage mit Angabe von FOR SYSTEM_TIME AS OF. Beispiel:

```
SELECT policy_id, coverage
   FROM policy_info
   FOR SYSTEM_TIME AS OF
      '2011-02-28-09.10.12.649592000000'
```

Diese Abfrage gibt drei Zeilen zurück. Von der Anweisung SELECT werden die Tabelle policy_info und die Tabelle hist_policy_info abgefragt. Die Beginnspalte eines Zeitraums ist inklusiv, die Endspalte ist exklusiv. Die Zeile der Protokolltabelle mit einem Spaltenwert für 'sys_end' von 2011-02-28-22.31.33.495925000000 entspricht *wert1*; der Wert muss aber kleiner als *wert1* sein, um zurückgegeben zu werden.

```
A123, 12000
C567, 25000
B345, 18000
```

- Abfrage mit Angabe von FOR SYSTEM_TIME FROM...TO. Beispiel:

```
SELECT policy_id, coverage, sys_start, sys_end
   FROM policy_info
   FOR SYSTEM_TIME FROM
      '0001-01-01-00.00.00.000000' TO '9999-12-30-00.00.00.000000000000'
   where policy_id = 'C567'
```

Diese Abfrage gibt zwei Zeilen zurück. Von der Anweisung SELECT wird die Tabelle policy_info und die Tabelle hist_policy_info abgefragt.

```
C567, 25000, 2011-02-28-09.10.12.649592000000, 9999-12-30-00.00.00.000000000000
C567, 20000, 2010-01-31-22.31.33.495925000000, 2011-02-28-09.10.12.649592000000
```

- Abfrage mit Angabe von FOR SYSTEM_TIME BETWEEN...AND. Beispiel:

```
SELECT policy_id, coverage
   FROM policy_info
   FOR SYSTEM_TIME BETWEEN
      '2011-02-28-09.10.12.649592000000' AND '9999-12-30-00.00.00.000000000000'
```

Diese Abfrage gibt drei Zeilen zurück. Von der Anweisung SELECT werden die Tabelle policy_info und die Tabelle hist_policy_info abgefragt. Die Zeilen mit einem Spaltenwert für 'sys_start' von '2011-02-28-09.10.12.649592000000' sind gleich *wert1* und werden zurückgegeben, weil die Anfangszeit eines Zeitraums enthalten ist. Die Zeilen mit einem Spaltenwert für 'sys_end' von '2011-02-28-09.10.12.649592000000 ' sind gleich *wert1* und werden nicht zurückgegeben, weil die Endzeit eines Zeitraums nicht enthalten ist.

```
A123, 12000
C567, 25000
B345, 18000
```

Weitere Beispiele

Der folgende Abschnitt enthält weitere Beispiele zur Abfrage von temporalen Tabellen für Systemzeitraum.

Abfrage mit anderen gültigen Datums-oder Zeitmarkenwerten

Die Tabelle policy_info wurde mit den als TIMESTAMP(12) deklarierten zeitbezogenen Spalten erstellt, sodass Abfragen mit anderen gültigen Datums- oder Zeitmarkenwerten vor der Ausführung so konvertiert werden, dass sie TIMESTAMP(12) verwenden. Beispiel:

```
SELECT policy_id, coverage
   FROM policy_info
   FOR SYSTEM_TIME AS OF '2011-02-28'
```

Diese Abfrage wird konvertiert und wie folgt ausgeführt:

```
SELECT policy_id, coverage
   FROM policy_info
   FOR SYSTEM_TIME AS OF '2011-02-28-00.00.00.000000000000'
```

Abfragen einer Sicht

Eine Sicht kann wie eine temporale Tabelle für Systemzeitraum abgefragt werden. FOR SYSTEM_TIME-Spezifikationen können nach dem Sichtverweis angegeben werden.

```
CREATE VIEW policy_2011(policy, start_date)
   AS SELECT policy_id, sys_start FROM policy_info;

SELECT * FROM policy_2011 FOR SYSTEM_TIME BETWEEN
   '2011-01-01-00.00.00.000000' AND '2011-12-31-23.59.59.999999999999';
```

Von der Anweisung SELECT wird die Tabelle `policy_info` und die Tabelle `hist_policy_info` für die Sicht `policy_2011` abgefragt. Daraufhin werden alle Richtlinien, die zu einem beliebigen Zeitpunkt im Jahr 2011 aktiv waren, zusammen mit dem Startdatum der Richtlinien zurückgegeben.

```
A123, 2010-01-31-22.31.33.495925000000
C567, 2011-02-28-09.10.12.649592000000
C567, 2010-01-31-22.31.33.495925000000
B345, 2010-01-31-22.31.33.495925000000
```

Wenn eine Sichtdefinition eine Zeitraumangabe enthält, können Abfragen für diese Sicht keine Zeitraumangaben enthalten. Die folgenden Anweisungen geben einen Fehler zurück, der auf mehrere Zeitraumangaben zurückzuführen ist:

```
CREATE VIEW all_policies AS SELECT * FROM policy_info;
   FOR SYSTEM_TIME AS OF '2011-02-28-09.10.12.649592000000';

SELECT * FROM all_policies FOR SYSTEM_TIME BETWEEN
   '2011-01-01-00.00.00.000000' AND '2011-12-31-23.59.59.999999999999';
```

Festlegen der Systemzeit für eine Sitzung

Durch Festlegen der Systemzeit mit dem Sonderregister SYSTEM_TIME werden die Änderungen, die bei der Ausführung einer Anwendung für bestimmte Zeitpunkte erforderlich sind, möglicherweise teilweise oder ganz überflüssig.

Informationen zu diesem Vorgang

Wenn Sie eine Ihrer Anwendungen für eine temporale Tabelle für Systemzeitraum ausführen wollen, um den Status Ihres Unternehmens zu einer Reihe verschiedener Daten abzufragen, können Sie das Datum in einem Sonderregister festlegen. Wenn Sie Ihre Daten mit Stand von heute, am Ende des letzten Quartals und an demselben Datum des Vorjahres abfragen müssen, ist es unter Umständen nicht möglich, die Anwendung zu ändern und AS OF-Spezifikationen zu den einzelnen SQL-Anweisungen hinzuzufügen. Diese Beschränkung ist bei der Verwendung von Paketanwendungen wahrscheinlich. Sie können bei solchen Szenarios das Datum mit dem Sonderregister CURRENT TEMPORAL SYSTEM_TIME auf Zeitmarken- oder Sitzungsebene festlegen.

Die Einstellung des Sonderregisters CURRENT TEMPORAL SYSTEM_TIME hat keine Auswirkungen auf reguläre Tabellen. Die im Sonderregister festgelegte Zeit wird nur von Abfragen für temporale Tabellen mit Versionssteuerung (temporale Tabellen für Systemzeitraum und bitemporale Tabellen) verwendet. Es gibt auch keine Auswirkungen auf DDL-Anweisungen. Das Sonderregister gilt nicht für Suchläufe, die für die Verarbeitung der referenziellen Integrität ausgeführt werden.
.

Wichtig: Wenn das Sonderregister CURRENT TEMPORAL SYSTEM_TIME auf einen Nichtnullwert eingestellt ist, werden Datenänderungsanweisungen wie IN-

SERT, UPDATE und DELETE für temporale Tabellen für Systemzeitraum blockiert. Wenn das Sonderregister auf einen Zeitpunkt in der Vergangenheit festgelegt wurde, beispielsweise vor fünf Jahren, bewirkt das Zulassen von Datenänderungsanweisungen möglicherweise Änderungen an Ihren Protokolldatensätzen. Dienstprogramme wie IMPORT und LOAD werden für temporale Tabellen für Systemzeitraum auch blockiert, wenn das Sonderregister CURRENT TEMPORAL SYSTEM_TIME auf einen Wert ungleich null gesetzt wurde.

Der Befehl BIND enthält die Option SYSTIMESENSITIVE, durch die angegeben wird, ob Verweise auf temporale Tabellen für Systemzeitraum in statischen und dynamischen SQL-Anweisungen durch den Wert des Sonderregisters CURRENT TEMPORAL SYSTEM_TIME betroffen sind. Verwenden Sie bei SQL-Prozeduren die Prozedur SET_ROUTINE_OPTS zum Definieren der mit dem Binden verwandten Optionen, der sogenannten Abfragecompilervariablen.

Vorgehensweise

Wenn dieses Sonderregister auf einen Wert ungleich null eingestellt wird, geben Anwendungen, die eine Abfrage absetzen, Daten zum Stand des betreffenden Datums oder der betreffenden Zeitmarke zurück. In den folgenden Beispielen werden Informationen aus den im Abschnitt Löschen von Daten aus einer temporalen Tabelle für Systemzeitraum aufgeführten Ergebnistabellen abgefragt.

- Legen Sie für das Sonderregister die aktuelle Zeitmarke fest und fragen Sie Daten von vor einem Jahr ab. Dabei wird als aktuelle Zeitmarke 2011-05-17-14.45.31.434235000000 angenommen:

```
SET CURRENT TEMPORAL SYSTEM_TIME = CURRENT TIMESTAMP – 1 YEAR;
SELECT policy_id, coverage FROM policy_info;
```

- Legen Sie für das Sonderregister eine Zeitmarke fest und verweisen Sie in den Sichtdefinitionen auf eine temporale Tabelle für Systemzeitraum.

```
CREATE VIEW view1 AS SELECT policy_id, coverage FROM policy_info;
CREATE VIEW view2 AS SELECT * FROM regular_table
   WHERE col1 IN (SELECT coverage FROM policy_info);
SET CURRENT TEMPORAL SYSTEM_TIME = TIMESTAMP '2011-02-28-09.10.12.649592000000';
SELECT * FROM view1;
SELECT * FROM view2;
```

- Legen Sie für das Sonderregister die aktuelle Zeitmarke fest und setzen Sie eine Abfrage ab, die eine Zeitraumspezifikation enthält. Dabei wird als aktuelle Zeitmarke 2011-05-17-14.45.31.434235000000 angenommen:

```
SET CURRENT TEMPORAL SYSTEM_TIME = CURRENT TIMESTAMP – 1 YEAR;
SELECT * FROM policy_info FOR SYSTEM_TIME AS OF '2011-02-28-09.10.12.649592000000';
```

Ergebnisse

Die Tabelle `policy_info` und die zugehörige Protokolltabelle sehen wie folgt aus:

Tabelle 44. Daten in der temporalen Tabelle für Systemzeitraum (policy_info) nach der DELETE-Anweisung

policy_id	coverage	sys_start	sys_end	ts_id
A123	12000	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
C567	25000	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000

Tabelle 45. Protokolltabelle (hist_policy_info) nach dem Löschvorgang

policy_id	coverage	sys_start	sys_end	ts_id
C567	20000	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.649592000000	2010-01-31-22.31.33.495925000000
B345	18000	2010-01-31-22.31.33.495925000000	2011-09-01-12.18.22.959254000000	2010-01-31-22.31.33.495925000000

- Bei der Anforderung von Daten von vor einem Jahr wird die Tabelle `policy_info` zum Stand von 2010-05-17-14.45.31.434235000000 abgefragt. Die Abfrage wird implizit wie folgt umgeschrieben:

  ```
  SELECT policy_id, coverage FROM policy_info
     FOR SYSTEM_TIME AS OF TIMESTAMP '2010-05-17-14.45.31.434235000000';
  ```

  Von der Anweisung SELECT wird die Tabelle `policy_info` und die Tabelle `hist_policy_info` abgefragt. Zurückgegeben wird Folgendes:

  ```
  A123, 12000
  C567, 20000
  B345, 18000
  ```

- Die Abfrage für `view1` wird implizit wie folgt umgeschrieben:

  ```
  SELECT * FROM view1 FOR SYSTEM_TIME AS OF CURRENT TEMPORAL SYSTEM_TIME;
  ```

  Dann wird die Abfrage wie folgt umgeschrieben:

  ```
  SELECT policy_id, coverage FROM policy_info
     FOR SYSTEM_TIME AS OF TIMESTAMP '2011-02-28-09.10.12.649592000000';
  ```

  Von der Anweisung SELECT wird die Tabelle `policy_info` und die Tabelle `hist_policy_info` abgefragt. Zurückgegeben wird Folgendes:

  ```
  A123, 12000
  C567, 25000
  B345, 18000
  ```

  Die Abfrage für `view2` umfasst eine Sicht für eine reguläre Tabelle, die auf eine temporale Tabelle für Systemzeitraum verweist. Dies bewirkt eine implizite Beziehung zwischen einer regulären Tabelle und dem Sonderregister. Die Abfrage wird implizit wie folgt umgeschrieben:

  ```
  SELECT * FROM view2 FOR SYSTEM_TIME AS OF CURRENT TEMPORAL SYSTEM_TIME;
  ```

  Dann wird die Abfrage wie folgt umgeschrieben:

  ```
  SELECT * FROM regular_table WHERE col1 in (SELECT coverage FROM policy_info
     FOR SYSTEM_TIME AS OF TIMESTAMP '2011-02-28-09.10.12.649592000000');
  ```

  Die Anweisung SELECT gibt Zeilen zurück, wobei die Werte für `col1` mit den Werten für `coverage` übereinstimmen.

- Da mehrere Zeitraumspezifikationen vorhanden sind, wird ein Fehler zurückgegeben. Für das Sonderregister wurde ein Nichtnullwert festgelegt und auch in der Abfrage wurde eine Zeit angegeben.

Löschen von temporaler Tabelle für Systemzeitraum

Beim Löschen einer temporalen Tabelle für Systemzeitraum werden auch die zugehörige Protokolltabelle und alle für die Protokolltabelle definierten Indizes gelöscht.

Vorbereitende Schritte

Wenn Sie eine temporale Tabelle für Systemzeitraum löschen wollen, müssen Sie zum Löschen der zugehörigen Protokolltabelle berechtigt sein.

Informationen zu diesem Vorgang

Eine Protokolltabelle wird implizit gelöscht, wenn die zugehörige temporale Tabelle für Systemzeitraum gelöscht wird. Eine Protokolltabelle kann nicht explizit mit der Anweisung DROP gelöscht werden.

Um den Verlust von Protokolldaten beim Löschen einer temporalen Tabelle für Systemzeitraum zu vermeiden, können Sie die Protokolltabelle mit dem Attribut RESTRICT ON DROP erstellen oder das Attribut RESTRICT ON DROP zu der erstellten Protokolltabelle hinzufügen. Wenn Sie versuchen, eine temporale Tabelle für Systemzeitraum zu löschen, deren Protokolltabelle das Attribut RESTRICT ON DROP aufweist, schlägt der Löschversuch der temporalen Tabelle für Systemzeitraum fehl (SQLSTATE 42893). Unterbrechen Sie in diesem Fall die Verknüpfung zwischen der temporalen Tabelle für Systemzeitraum und der Protokolltabelle, indem Sie das Attribut VERSIONING entfernen, und führen Sie die Anweisung DROP noch einmal aus.

Wenn das Attribut VERSIONING aus der Tabelle gelöscht wird, werden alle Pakete mit der Versionierungsabhängigkeit für die Tabelle ungültig gemacht. Andere abhängige Objekte wie Sichten oder Trigger werden im Systemkatalog mit 'N' als ungültig markiert. Es wird eine automatische Reaktivierung ausgeführt. Die Objekte, deren Reaktivierung fehlschlägt, verbleiben als ungültig im Katalog. Bestimmte Objekte können nur durch eine explizite Benutzeraktion gültig gemacht werden.

Vorgehensweise

Gehen Sie wie folgt vor, um eine temporale Tabelle für Systemzeitraum und die zugehörige Protokolltabelle zu löschen:

1. Optional: Schützen Sie Protokolldaten vor der Löschung:
   a. Wenn die Protokolltabelle nicht mit dem Attribut RESTRICT ON DROP erstellt wurde, ändern Sie die Protokolltabelle, indem Sie das Attribut RESTRICT ON DROP festlegen. Wenn beispielsweise das Protokoll von Versicherungspolicen aufgrund von Prüfvorschriften aufbewahrt werden musste, muss die Protokolltabelle geschützt werden.
      ```
      ALTER TABLE hist_policy_info ADD RESTRICT ON DROP;
      ```
   b. Unterbrechen Sie die Verknüpfung zwischen der temporalen Tabelle für Systemzeitraum und einer Protokolltabelle mit dem Attribut RESTRICT ON DROP, indem Sie das Attribut VERSIONING entfernen. Beispiel:
      ```
      ALTER TABLE policy_info DROP VERSIONING;
      ```
2. Löschen Sie die temporale Tabelle für Systemzeitraum mit der Anweisung DROP. Beispiel: Die Tabellen für Versicherungspolicen, die in dem Beispiel im Abschnitt Erstellen einer temporalen Tabelle für Systemzeitraum erstellt wurden, sind nicht mehr erforderlich.
   ```
   DROP TABLE policy_info;
   ```

Ergebnisse

Die vorherigen Befehle haben wie folgt Auswirkungen auf die Tabellen `policy_info` und `hist_policy_info`.

- Die Anweisung DROP löscht die temporale Tabelle für Systemzeitraum explizit und die zugehörige Protokolltabelle implizit. Die Tabellen `policy_info` und `hist_policy_info` werden gelöscht. Die Objekte, die direkt oder indirekt von diesen Tabellen abhängig sind, werden gelöscht oder funktionsunfähig gemacht.
- Nachdem das Attribut RESTRICT ON DROP zu der Protokolltabelle zugeordnet wurde, schlagen alle Versuche, die Tabelle `policy_info` zu löschen, fehl (SQLSTATE 42893). Eine temporale Tabelle für Systemzeitraum kann auch so erstellt oder geändert werden, dass das Attribut RESTRICT ON DROP verwendet wird.
- Nachdem die Verknüpfung zwischen der temporalen Tabelle für Systemzeitraum unterbrochen wurde, kann die Tabelle `policy_info` gelöscht werden, wobei die Protokolltabelle `hist_policy_info` erhalten bleibt.

Löschen von Tabellenbereichen

Wenn ein Tabellenbereich eine Protokolltabelle enthält, jedoch nicht die zugeordnete temporale Tabelle für Systemzeitraum, kann dieser Tabellenbereich nicht explizit gelöscht werden. Beispiel: Mit den Tabellen für Versicherungspolicen, die in den Tabellenbereichen `policy_space` und `hist_space` erstellt wurden, wird die folgende Anweisung blockiert:

```
DROP TABLESPACE hist_space;
```

Wenn der Tabellenbereich, der eine Protokolltabelle enthält, und der Tabellenbereich, der die zugehörige temporale Tabelle für Systemzeitraum enthält, gemeinsam eingebunden werden, ist die Anweisung zulässig. Die folgende Anweisung beispielsweise wird erfolgreich ausgeführt:

```
DROP TABLESPACE policy_space hist_space;
```

Eine Protokolltabelle wird implizit gelöscht, wenn der Tabellenbereich der zugehörigen temporalen Tabelle für Systemzeitraum gelöscht wird. Mit der folgenden Anweisung wird beispielsweise die Protokolltabelle `hist_policy_info` gelöscht:

```
DROP TABLESPACE policy_space;
```

Dienstprogramme und Tools

Es stehen eine Reihe von Tools und Dienstprogrammen zur Verfügung, mit deren Hilfe Sie mit den Daten in den temporale Tabellen arbeiten und diese verwalten können.

Zum Arbeiten mit temporalen Tabellen und zum Verwalten dieser Tabellen stehen die folgenden Tools zur Verfügung:

- Daten importieren (siehe „IMPORT")
- Daten laden (siehe „LOAD" auf Seite 408)
- Onlinetabellenversetzung (siehe „Prozedur ADMIN_MOVE_TABLE" auf Seite 409)
- Quiesce für einen Tabellenbereich durchführen (siehe „Befehl QUIESCE TABLESPACES FOR TABLE" auf Seite 409)
- Replikation (siehe „Replikation" auf Seite 410)
- Aktualisierende Recovery (siehe „Aktualisierende Recovery" auf Seite 410)
- Prozedur ADMIN_COPY_SCHEMA (siehe ADMIN_COPY_SCHEMA)

IMPORT

Beim Importieren von Daten in temporale Tabellen für Systemzeitraum werden Dateitypmodifikatoren verwendet, um die Inhalte in der externen Datei zu ignorie-

ren, die möglicherweise auf die vom Datenbankmanager generierten Spalten in der temporale Tabelle für Systemzeitraum angewendet werden. Beim Importieren von Daten in eine temporale Tabelle für Systemzeitraum stehen die folgenden Modifikatoren zur Verfügung:

periodignore
Mit diesem Modifikator wird das Importdienstprogramm darüber informiert, dass die Daten für die Spalten des Zeitraums SYSTEM_TIME in der externen Datei vorhanden sind, jedoch ignoriert werden sollen. Bei Angabe dieses Modifikators werden alle Spaltenwerte für einen Zeitraum vom Dienstprogramm generiert.

periodmissing
Mit diesem Modifikator wird das Importdienstprogramm darauf hingewiesen, dass die externe Datendatei keine Daten für die Spalten des Zeitraums SYSTEM_TIME enthält. Bei Angabe dieses Modifikators werden alle Spaltenwerte für einen Zeitraum vom Dienstprogramm generiert.

transactionidignore
Mit diesem Modifikator wird das Importdienstprogramm darüber informiert, dass die Daten für die Spalte mit der Transaktionsstart-ID in der externen Datei vorhanden sind, jedoch ignoriert werden sollen. Bei Angabe dieses Modifikators wird der Wert für die Spalte mit der Transaktionsstart-ID vom Dienstprogramm generiert.

transactionidmissing
Mit diesem Modifikator wird das Importdienstprogramm darauf hingewiesen, dass die externe Datendatei keine Daten für die Spalte mit der Transaktionsstart-ID enthält. Bei Angabe dieses Modifikators wird der Wert für die Spalte mit der Transaktionsstart-ID vom Dienstprogramm generiert.

Im Gegensatz zum Ladedienstprogramm verfügt das Importdienstprogramm über keine Modifikatoren zum Überschreiben der GENERATED ALWAYS-Spalten.

LOAD

Beim Laden von Daten in temporale Tabellen für Systemzeitraum werden Dateitypmodifikatoren verwendet, um die Daten in der externen Datei zu ignorieren, die möglicherweise auf die vom Datenbankmanager generierten Spalten angewendet werden, oder um vom Benutzer bereitgestellte Werte in diese generierten Spalten zu laden. Beim Laden von Daten in eine temporale Tabelle für Systemzeitraum stehen die folgenden Modifikatoren zur Verfügung. LOAD REPLACE ist bei temporalen Tabellen für Systemzeitraum blockiert.

periodignore
Mit diesem Modifikator wird das Ladedienstprogramm darüber informiert, dass die Daten für die Spalten des Zeitraums SYSTEM_TIME in der externen Datei vorhanden sind, jedoch ignoriert werden sollen. Bei Angabe dieses Modifikators werden alle Spaltenwerte für einen Zeitraum vom Dienstprogramm generiert.

periodmissing
Mit diesem Modifikator wird das Ladedienstprogramm darauf hingewiesen, dass die externe Datendatei keine Daten für die Spalten des Zeitraums SYSTEM_TIME enthält. Bei Angabe dieses Modifikators werden alle Spaltenwerte für einen Zeitraum vom Dienstprogramm generiert.

periodoverride
Mit diesem Modifikator wird das Ladedienstprogramm angewiesen, vom

Benutzer bereitgestellte Werte für die Zeilenbeginn- und -endspalten des Zeitraums SYSTEM_TIME zu akzeptieren. Dieser Modifikator überschreibt die Klausel GENERATED ALWAYS. Dieser Modifikator kann bei der Verwaltung von Protokolldaten oder beim Laden von Daten mit Zeitmarken in eine temporale Tabelle für Systemzeitraum nützlich sein. Bei Verwendung dieses Modifikators werden alle Zeilen ohne Daten oder mit Nulldaten in den Zeilenbeginn- und -endspalten zurückgewiesen.

transactionidignore
Mit diesem Modifikator wird das Ladedienstprogramm darüber informiert, dass die Daten für die Spalte mit der Transaktionsstart-ID in der externen Datei vorhanden sind, jedoch ignoriert werden sollen. Bei Angabe dieses Modifikators wird der Wert für die Spalte mit der Transaktionsstart-ID vom Dienstprogramm generiert.

transactionidmissing
Mit diesem Modifikator wird das Ladedienstprogramm darauf hingewiesen, dass die externe Datendatei keine Daten für die Spalte mit der Transaktionsstart-ID enthält. Bei Angabe dieses Modifikators wird der Wert für die Spalte mit der Transaktionsstart-ID vom Dienstprogramm generiert.

transactionidoverride
Mit diesem Modifikator wird das Ladedienstprogramm angewiesen, vom Benutzer bereitgestellte Werte für die Spalte mit der Transaktionsstart-ID zu akzeptieren. Dieser Modifikator überschreibt die Klausel GENERATED ALWAYS. Bei Verwendung dieses Modifikators werden alle Zeilen ohne Daten oder mit Nulldaten in der Spalte für die Transaktionsstart-ID zurückgewiesen.

Prozedur ADMIN_MOVE_TABLE

Wenn Sie die gespeicherte Prozedur ADMIN_MOVE_TABLE verwenden, um Daten in einer aktiven temporalen Tabelle für Systemzeitraum in eine neue Tabelle mit demselben Namen zu versetzen, werden die folgenden Aktionen blockiert.

- ALTER TABLE-Operationen, die die Definition der temporalen Tabelle für Systemzeitraum oder der zugehörigen Protokolltabelle ändern, werden während einer Onlineversetzungsoperation blockiert.
- Die Option KEEP von ADMIN_MOVE_TABLE steht für temporale Tabellen für Systemzeitraum nicht zur Verfügung.

Die Onlineversetzungsoperation wird für Protokolltabellen nicht unterstützt.

Befehl QUIESCE TABLESPACES FOR TABLE

Bei Ausführung des Befehls QUIESCE TABLESPACES FOR TABLE für eine temporale Tabelle für Systemzeitraum werden alle Tabellenbereiche, die dieser temporalen Tabelle für Systemzeitraum und der zugehörigen Protokolltabelle zugeordnet sind, in den Quiescemodus versetzt. Bei Ausführung des Befehls für eine Protokolltabelle werden alle Tabellenbereiche, die dieser Protokolltabelle und der zugehörigen temporalen Tabelle für Systemzeitraum zugeordnet sind, in den Quiescemodus versetzt.

Replikation

Beim Replizieren einer temporalen Tabelle für Systemzeitraum können Spalten mit den folgenden generierten Attributen nicht bei der Replikation berücksichtigt werden, wenn es sich bei der Zieltabelle um eine andere temporale Tabelle für Systemzeitraum handelt:

- GENERATED ALWAYS AS ROW BEGIN
- GENERATED ALWAYS AS ROW END
- GENERATED ALWAYS AS TRANSACTION START ID

Aktualisierende Recovery

Wenn der Tabellenbereich für eine temporale Tabelle für Systemzeitraum oder eine bitemporale Tabelle bis zu einem bestimmten Zeitpunkt aktualisierend wiederhergestellt wird, muss auch der Tabellenbereich für die zugehörige Protokolltabelle in derselben ROLLFORWARD-Anweisung bis zum selben Zeitpunkt aktualisierend wiederhergestellt werden. Ebenso gilt Folgendes: Wenn der Tabellenbereich für eine Protokolltabelle bis zu einem bestimmten Zeitpunkt aktualisierend wiederhergestellt wird, muss auch der Tabellenbereich für die temporale Tabelle für Systemzeitraum oder eine bitemporale Tabelle bis zum selben Zeitpunkt aktualisierend wiederhergestellt werden. Der Tabellenbereich für die temporale Tabelle für Systemzeitraum und der Tabellenbereich für die Protokolltabelle können jedoch auch einzeln bis zum Ende der Protokolle wiederhergestellt werden.

Prozedur ADMIN_COPY_SCHEMA

Mit der Prozedur ADMIN_COPY_SCHEMA kann ein bestimmtes Schema mit allen enthaltenen Objekten kopiert werden. Die neuen Zielschemaobjekte werden mit den gleichen Objektnamen wie die Objekte im Quellenschema, jedoch mit dem Qualifikationsmerkmal 'target' erstellt. Das Verwenden der Prozedur ADMIN_COPY_SCHEMA für temporale Tabellen für Systemzeitraum wird unterstützt. Bei der Prozedur wird vorausgesetzt, dass die temporale Tabelle für Systemzeitraum und die Protokolltabelle in demselben Schema enthalten sind. Ist dies nicht der Fall, wird keine Tabelle kopiert und es wird ein Fehler aufgezeichnet.

Schemaänderungen

Um die Integrität der Beziehung zwischen der temporalen Tabelle für Systemzeitraum und der dazugehörigen Protokolltabelle zu wahren, sind nur bestimmte Änderungen am Schema einer temporalen Tabelle für Systemzeitraum zulässig. Alle Änderungen, die zu Datenverlusten führen würden, werden nicht zugelassen.

Sie können die folgenden Änderungen in Ihrer temporalen Tabelle für Systemzeitraum vornehmen. Diese Änderungen werden implizit in die zugehörige Protokolltabelle übertragen, wenn Sie über die entsprechenden Berechtigungen verfügen. Diese Änderungen können nicht explizit in der Protokolltabelle vorgenommen werden.

- ADD COLUMN (ausgenommen generierte Spalten)
- RENAME COLUMN
- ALTER COLUMN (in den Fällen, in denen keine Protokolldaten verloren gegangen sind). Beispielsweise ist die Änderung des Datentyps einer Spalte von VARCHAR(50) in VARCHAR(100) oder von INTEGER in DECIMAL zulässig. Die umgekehrte Änderung von VARCHAR(100) in VARCHAR(50) oder von DECIMAL in INTEGER wird jedoch blockiert, da dadurch die Länge oder die Genauigkeit einer Spalte verringert und möglicherweise ein Datenverlust verursacht würde.

Die folgenden Änderungen an Ihren temporalen Tabellen für Systemzeitraum sind nicht zulässig, weil dadurch Datenverluste entstehen können:

- DROP COLUMN
- ADD COLUMN (generierte)
- ALTER COLUMN (in den Fällen, in denen möglicherweise Protokolldaten verloren gegangen sind). Beispielsweise ist die Änderung des Datentyps einer Spalte von VARCHAR(50) in VARCHAR(100) oder von INTEGER in DECIMAL zulässig. Die umgekehrte Änderung von VARCHAR(100) in VARCHAR(50) oder von DECIMAL in INTEGER wird jedoch blockiert, da dadurch die Länge oder die Genauigkeit einer Spalte verringert würde.

Die Versionierung verknüpft Ihre temporalen Tabellen für Systemzeitraum mit der zugehörigen Protokolltabelle. Wenn die Versionierung aktiviert ist, werden bestimmte ALTER TABLE-Operationen für temporale Tabellen für Systemzeitraum und für Protokolltabellen geblockt.

- ALTER TABLE DROP PERIOD
- ALTER TABLE ADD MATERIALIZED
- ALTER TABLE ACTIVATE NOT LOGGED INITIALLY
- ALTER TABLE ADD SECURITY POLICY
- ALTER TABLE DROP SECURITY POLICY
- ALTER TABLE SECURED WITH ALTER

Die in der vorangegangenen Liste nicht aufgeführten ALTER TABLE-Operationen werden unterstützt, aber sie werden nicht aus der temporalen Tabelle für Systemzeitraum in die zugehörige Protokolltabelle repliziert.

Außerdem werden RENAME INDEX- und RENAME TABLE-Operationen unterstützt, jedoch nicht aus der temporalen Tabelle für Systemzeitraum in die zugehörige Protokolltabelle repliziert.

Cursor und temporale Tabellen für Systemzeitraum

Cursor, die zum Aktualisieren oder Löschen von Zeilen für eine Abfrage verwendet werden, die möglicherweise auf Zeilen in einer Protokolltabelle verweist, müssen schreibgeschützt sein.

Im folgenden Beispiel wird die Anweisung erfolgreich ausgeführt, weil der Cursor `appcur` schreibgeschützt ist.

```
DECLARE appcur CURSOR FOR SELECT * FROM policy_info FOR SYSTEM_TIME AS OF '2011-02-28';
```

Die folgende Anweisung verursacht jedoch einen Fehler, weil die Cursorverweise auf die Protokollzeilen nicht schreibgeschützt sind:

```
DECLARE appcur CURSOR FOR SELECT * FROM policy_info FOR SYSTEM_TIME
AS OF '2011-02-28' FOR UPDATE;
```

Bereichspartitionierung und temporale Tabellen für Systemzeitraum

Die Tabellendaten einer temporalen Tabelle für Systemzeitraum können auf mehrere Speicherobjekte, d. h. Datenpartitionen oder Datenbereiche, verteilt sein. Auch die einer temporalen Tabelle für Systemzeitraum zugeordnete Protokolltabelle kann bereichspartitioniert sein.

Wenn die Versionssteuerung aktiviert ist, wird beim Zuordnen einer Partition zu einer temporalen Tabelle für Systemzeitraum bzw. beim Aufheben einer Partitionszuordnung bei einer temporalen Tabelle für Systemzeitraum das folgende Verhalten angewendet:

Partitionen zuordnen

- Einer temporalen Tabelle für Systemzeitraum kann eine Tabelle zugeordnet werden, wenn die Versionssteuerung aktiviert ist.
- Die zuzuordnende Tabelle muss die drei Zeitmarkenspalten (ROW BEGIN, ROW END und TRANSACTION START ID) enthalten. Diese Zeitmarkenspalten müssen dieselben Definitionen aufweisen wie die Spalten in der temporalen Tabelle für Systemzeitraum.
- Die Zeitraumdefinition für SYSTEM_TIME ist bei der zuzuordnenden Tabelle nicht erforderlich.
- Bei aktivierter Versionssteuerung kann die Anweisung 'SET INTEGRITY ... FOR EXCEPTION' nicht ausgeführt werden, da das Versetzen von Ausnahmezeilen in eine Ausnahmetabelle zu Protokollierungslücken führt. Da die Ausnahmezeilen nicht in der Protokolltabelle aufgezeichnet werden, ist die Überprüfbarkeit der Daten in Ihrer temporalen Tabelle für Systemzeitraum und der zugehörigen Protokolltabelle gefährdet. Sie können die Versionssteuerung bei Bedarf vorübergehend stoppen, die Anweisung 'SET INTEGRITY ... FOR EXCEPTION' ausführen und die Versionssteuerung anschließend erneut aktivieren.

Zuordnung von Partitionen aufheben

- Die Zuordnung einer Tabelle zu einer temporalen Tabelle für Systemzeitraum kann nicht aufgehoben werden, während die Versionssteuerung aktiviert ist. Sie können die Versionssteuerung bei Bedarf stoppen und anschließend die Zuordnung der jeweiligen Partition zur Basistabelle aufheben. Die abgehängte Partition wird zu einer unabhängigen Tabelle. Bei einer Protokolltabelle ist das Stoppen der Versionssteuerung vor dem Aufheben einer Partitionszuordnung nicht erforderlich.
- Eine abgehängte Partition behält die drei Zeitmarkenspalten (ROW BEGIN, ROW END und TRANSACTION START ID) bei, die Zeitraumdefinition für SYSTEM_TIME jedoch nicht.
- Die Zeilen einer abgehängten Partition werden nicht automatisch in die Protokolltabelle versetzt. Wenn die Historie aufrechterhalten werden soll, müssen die Zeilen manuell versetzt werden. Wenn Sie die Zeilen manuell in die Protokolltabelle versetzen, müssen Sie die Zeitmarke ROW END an den Zeitpunkt anpassen, zu dem die Zeilen von aktuellen Zeilen zu Protokollzeilen wurden. Ohne diese Änderungen können zeitbezogene Abfragen zu unerwarteten Ergebnissen führen.

Datenzugriffssteuerung für temporale Tabellen für Systemzeitraum

Die zeilen- und spaltenbezogene Zugriffssteuerung kann sowohl für eine temporale Tabelle für Systemzeitraum als auch für die zugehörige Protokolltabelle definiert werden.

Die zeilen- und spaltenbezogene Zugriffssteuerung (Row and Column Access Control, RCAC) ist eine Datensicherheitsstufe, bei der der Zugriff auf eine Tabelle auf Zeilenebene und/oder Spaltenebene gesteuert wird. RCAC kann auf temporale Tabellen für Systemzeitraum und auf Protokolltabellen angewendet werden. Wenn RCAC nur für eine temporale Tabelle für Systemzeitraum aktiviert wird, aktiviert

der Datenbankmanager automatisch die zeilenbezogene Zugriffssteuerung für die zugehörige Protokolltabelle und erstellt eine Standardzeilenberechtigung für die Protokolltabelle.

Auch weenn die Protokolltabelle über die Standardzeilenberechtigung geschützt ist, führen Aktualisierungen und Löschvorgänge zur Generierung von Protokollzeilen in der Protokolltabelle. Beim Ausführen einer AS OF-Abfrage für eine temporale Tabelle für Systemzeitraum werden die RCAC-Zeilenberechtigungen und -Spaltenmasken für die temporale Tabelle für Systemzeitraum auch auf die von der Protokolltabelle zurückgegebenen Zeilen angewendet.

Beim direkten Zugriff auf eine Protokolltabelle werden alle für die Protokolltabelle definierten Zeilen- und Spaltenregeln angewendet.

Einschränkungen für temporale Tabellen für Systemzeitraum

Für temporale Tabellen für Systemzeitraum gelten bestimmte Einschränkungen. Diese Einschränkungen können sich auf Ihre Implementierung von temporalen Tabellen für Systemzeitraum auswirken.

Die folgenden Einschränkungen gelten für temporale Tabellen für Systemzeitraum:

- Kennsatzbasierte Zugriffssteuerung (Label-Based Access Control, LBAC) wird für temporale Tabellen für Systemzeitraum nicht unterstützt. Solange die Versionierung der Daten für Systemzeitraum aktiviert ist, wird das Hinzufügen von Zeilen- und Spaltenkennsätzen sowohl in einer temporalen Tabelle für Systemzeitraum als auch in einer Protokolltabelle blockiert. Wenn die Versionierung mit einer Anweisung ALTER TABLE aktiviert ist, stellt der Datenbankmanager sicher, dass weder in der temporalen Tabelle für Systemzeitraum noch in der Protokolltabelle Zeilen oder Spalten durch Kennsätze geschützt sind.
- Operationen vom Typ ALTER, die Datenverluste verursachen können, werden in temporalen Tabellen für Systemzeitraum nicht unterstützt.
- Anweisungen wie ALTER TABLE ACTIVATE NOT LOGGED INITIALLY werden für die temporale Tabelle für Systemzeitraum und für die Protokolltabelle blockiert.
- Eine temporale Tabelle für Systemzeitraum kann nicht in eine MQT (Materialized Query Table) umgewandelt werden.
- Dienstprogramme zum Löschen von Daten in temporalen Tabellen für Systemzeitraum werden blockiert (dies gilt auch für LOAD REPLACE und IMPORT REPLACE).
- Die Anweisung TRUNCATE wird in einer temporalen Tabelle für Systemzeitraum nicht unterstützt.
- Die folgenden Operationen für Schemaänderung werden in temporalen Tabellen für Systemzeitraum nicht unterstützt:
  - ALTER TABLE DROP COLUMN
  - ALTER TABLE ALTER COL (das Ändern von Zeichenfolgedatentypen in einen Typ, der das Abschneiden von Daten erfordert) wird nicht unterstützt. Das Ändern numerischer Datentypen in einen Datentyp mit niedrigerer Genauigkeit wird ebenfalls nicht unterstützt.
  - ALTER TABLE ADD GENERATED COLUMN
- Wenn Sie einen Tabellenbereich, der die temporale Tabelle für Systemzeitraum enthält, bis zu einem bestimmten Zeitpunkt aktualisierend wiederherstellen, müssen Sie auch den Tabellenbereich, der die zugehörige Protokolltabelle enthält, bis zu demselben Zeitpunkt als Gruppe aktualisierend wiederherstellen. Ebenso gilt Folgendes: Wenn der Tabellenbereich für eine Protokolltabelle bis zu

einem bestimmten Zeitpunkt aktualisierend wiederhergestellt wird, muss auch der Tabellenbereich für die temporale Tabelle für Systemzeitraum oder eine bitemporale Tabelle bis zum selben Zeitpunkt aktualisierend wiederhergestellt werden. Die aktualisierende Wiederherstellung des Tabellenbereichs der temporalen Tabellen für Systemzeitraum (oder des Tabellenbereichs der Protokolltabellen) bis zum Ende der Protokolle ist jedoch zulässig.

- Sie können zwar einen Kurznamen für eine temporale Tabelle für Systemzeitraum erstellen, aber die temporalen Informationen werden dadurch nicht zugänglich und temporale Operationen über Kurznamen werden nicht unterstützt. Beispielsweise werden Definitionsoperationen für temporale Daten und temporale Abfragen für föderierte Kurznamen blockiert.
- IMPORT- und LOAD-Operationen in temporale Tabellen für Systemzeitraum werden blockiert, wenn das Sonderregister CURRENT TEMPORAL SYSTEM_TIME auf einen Nullwert gesetzt ist.

Temporale Tabellen für Anwendungszeitraum

Bei einer temporalen Tabelle für Anwendungszeitraum handelt es sich um eine Tabelle, die den Aspekt der zeitgebundenen Gültigkeit von Anwendungsdaten speichert. Mit einer temporalen Tabelle für Anwendungszeitraum können Sie Daten in Abhängigkeit von Zeitbedingungen verwalten, indem Sie Zeiträume für die Gültigkeit der Daten definieren.

Ähnlich einer temporalen Tabelle für Systemzeitraum enthält auch eine temporale Tabelle für Anwendungszeitraum enthält einen Zeitraum für BUSINESS_TIME mit Spalten, die angeben, in welchem Zeitraum die Daten in der betreffenden Zeile gültig sind. Sie geben eine Zeit für den Anfang und für das Ende des den einzelnen Zeilen zugeordneten Zeitraums für BUSINESS_TIME an. Anders als bei einer temporalen Tabelle für Systemzeitraum gibt es jedoch keine separate Protokolltabelle. Auf die Vergangenheit, Gegenwart oder Zukunft bezogene Datumsangaben werden mit den zugehörigen Geschäftsdaten in einer einzelnen Tabelle verwaltet. Sie können Datenwerte über den Zeitraum für BUSINESS_TIME steuern und mithilfe von temporalen Tabellen für Anwendungszeitraum vergangene, gegenwärtige und zukünftige Datenstände modellieren.

Zeitraum für BUSINESS_TIME

Die Zeitraumspalten für BUSINESS_TIME in einer temporalen Tabelle für Anwendungszeitraum erfassen, wann die Version einer Zeile aus der Perspektive eines Benutzers oder einer Geschäftsanwendung gültig ist.

Der Zeitraum für BUSINESS_TIME enthält zwei DATE- oder TIMESTAMP(*p*)-Spalten (dabei kann *p* einen Wert von 0 bis 12 annehmen). Diese Spalten werden von Ihnen oder von einer Geschäftsanwendung gefüllt. Die beiden Spalten in einem Zeitraum für BUSINESS_TIME geben den Anfang und das Ende des Gültigkeitszeitraums an. Anders als bei diesen Spalten werden die Werte für die Zeitraumspalten für SYSTEM_TIME vom Datenbankmanager generiert. Die Zeitraumspalten für BUSINESS_TIME müssen als NOT NULL definiert werden und dürfen keine generierten Spalten sein.

Die Zeiträume für BUSINESS_TIME sind inklusiv-exklusiv: Die Zeit für den Anfang des Gültigkeitszeitraums ist in BUSINESS_TIME eingeschlossen, die Endzeit nicht.

Wenn ein Zeitraum für BUSINESS_TIME in einer Tabelle definiert wird, wird auch eine implizite Prüfung auf Integritätsbedingung mit der Bezeichnung DB2_GENERATED_CHECK_CONSTRAINT_FOR_BUSINESS_TIME generiert, die

gewährleistet, dass der Wert für das Ende des Gültigkeitszeitraums größer ist als der Wert für den Start des Gültigkeitszeitraums. Wenn bereits eine Integritätsbedingung mit dieser Bezeichnung vorhanden ist, wird ein Fehler zurückgegeben. Diese Integritätsbedingung ist hilfreich für die Unterstützung von Operationen, die explizit Daten in diese Spalten eingeben (z. B. Einfüge- oder Ladeoperationen).

Eine temporale Tabelle für Anwendungszeitraum kann so definiert werden, dass Zeilen mit demselben Schlüssel keine sich überschneidenden Zeiträume für BUSINESS_TIME enthalten. Diese Einschränkung kann beispielsweise verhindern, dass zwei Versionen einer Versicherungspolice gleichzeitig gültig sind. Dieser Kontrollmechanismus kann genutzt werden, indem eine Klausel BUSINESS_TIME WITHOUT OVERLAPS zu einer Anweisung für einen Primärschlüssel, für eine eindeutige Integritätsbedingung oder für die Erstellung eines eindeutigen Indexes hinzugefügt wird. Die Umsetzung erfolgt durch den Datenbankmanager. Der Einsatz dieses Kontrollmechanismus ist optional.

Erstellen einer temporalen Tabelle für Anwendungszeitraum

Die Erstellung einer temporalen Tabelle für Anwendungszeitraum resultiert in einer Tabelle, die Daten ausgerichtet an dem Zeitpunkt, zu dem die Daten gültig bzw. wirksam sind, verwaltet.

Informationen zu diesem Vorgang

Fügen Sie bei der Erstellung einer temporalen Tabelle für Anwendungszeitraum einen BUSINESS_TIME-Zeitraum ein, der angibt, wann die Daten in einer Zeile gültig sind. Optional kann festgelegt werden, dass sich überschneidende Zeiträume von BUSINESS_TIME nicht zulässig sind und die Werte in Bezug auf alle Zeiträume eindeutig sein sollen. Das Beispiel im folgenden Abschnitt zeigt die Erstellung einer Tabelle, in der die Daten von Versicherungspolicen für die Kunden einer Versicherungsgesellschaft gespeichert werden.

Vorgehensweise

Gehen Sie wie folgt vor, um eine temporale Tabelle für Anwendungszeitraum zu erstellen:

1. Erstellen Sie eine Tabelle mit einem BUSINESS_TIME-Zeitraum. Beispiel:

```
CREATE TABLE policy_info
(
 policy_id       CHAR(4) NOT NULL,
 coverage        INT NOT NULL,
 bus_start       DATE NOT NULL,
 bus_end         DATE NOT NULL,
 PERIOD BUSINESS_TIME (bus_start, bus_end)
 );
```

2. Optional: Erstellen Sie einen eindeutigen Index, der sich überschneidende Zeiträume von BUSINESS_TIME für dieselbe Police `policy_id` verhindert. Beispiel:

```
CREATE UNIQUE INDEX ix_policy
   ON policy_info (policy_id, BUSINESS_TIME WITHOUT OVERLAPS);
```

Ergebnisse

In der Tabelle `policy_info` wird der Versicherungsdeckungswert für einen Kunden gespeichert. Die auf den Zeitraum BUSINESS_TIME bezogenen Spalten (`bus_start` und `bus_end`) geben an, ob der Wert für die Versicherungsdeckung gültig ist.

Tabelle 46. Erstellte temporale Tabelle für Anwendungszeitraum (policy_info)

policy_id	coverage	bus_start	bus_end

Der Index ix_policy mit BUSINESS_TIME WITHOUT OVERLAPS als letzter Spalte in der Spaltenliste für Indexschlüssel stellt sicher, dass für die Versicherungsdeckungswerte der Kunden keine sich überschneidenden Zeiträume vorhanden sind.

Beispiel

Der folgende Abschnitt enthält weitere Beispiele zur Erstellung von temporalen Tabellen für Anwendungszeitraum.

Ändern einer vorhandenen Tabelle in eine temporale Tabelle für Anwendungszeitraum

Das folgende Beispiel fügt Zeitspalten und einen Zeitraum BUSINESS_TIME zu einer vorhandenen Tabelle (employees) hinzu und aktiviert die Funktionalität von temporalen Tabellen für Anwendungszeitraum. Das Hinzufügen der Klausel BUSINESS_TIME WITHOUT OVERLAPS stellt sicher, dass ein Mitarbeiter in einem bestimmten Zeitraum nur einmal aufgelistet wird.

```
ALTER TABLE employees ADD COLUMN bus_start DATE NOT NULL;
ALTER TABLE employees ADD COLUMN bus_end DATE NOT NULL;
ALTER TABLE employees ADD PERIOD BUSINESS_TIME(bus_start, bus_end);
ALTER TABLE employees ADD CONSTRAINT uniq
   UNIQUE(employee_id, BUSINESS_TIME WITHOUT OVERLAPS);
```

Verhindern sich überschneidender Zeiträume

Im Abschnitt „Vorgehensweise" stellt ein Index sicher, dass keine sich überschneidenden BUSINESS_TIME-Zeiträume vorhanden sind. Im folgenden alternativen Beispiel wird bei der Erstellung der Tabelle policy_info eine PRIMARY KEY-Deklaration verwendet, die sicherstellt, dass sich überschneidende Zeiträume von BUSINESS_TIME nicht zulässig sind. Dies bedeutet, dass zu keinem Zeitpunkt zwei Versionen einer Police gleichzeitig gültig sein können.

```
CREATE TABLE policy_info
(
  policy_id CHAR(4) NOT NULL,
  coverage INT NOT NULL,
  bus_start DATE NOT NULL,
  bus_end DATE NOT NULL,
  PERIOD BUSINESS_TIME(bus_start, bus_end),
  PRIMARY KEY(policy_id, BUSINESS_TIME WITHOUT OVERLAPS)
);
```

Sicherstellen der Eindeutigkeit von Zeiträumen

Bei dem folgenden Beispiel wird eine Tabelle namens product_availability erstellt, über die eine Firma die Produkte verfolgt, die sie vertreibt, sowie die Lieferanten dieser Produkte und die Preise, die diese Lieferanten berechnen. Mehrere Lieferanten können gleichzeitig dasselbe Produkt liefern, eine PRIMARY KEY-Deklaration stellt jedoch sicher, dass jeder Lieferant zu einem bestimmten Zeitpunkt nur einen bestimmten Preis in Rechnung stellen kann.

```
CREATE TABLE product_availability
(
  product_id CHAR(4) NOT NULL,
  supplier_id INT NOT NULL,
  product_price DECIMAL NOT NULL
```

```
  bus_start DATE NOT NULL,
  bus_end DATE NOT NULL,
  PERIOD BUSINESS_TIME(bus_start, bus_end),
  PRIMARY KEY(product_id, supplier_id, BUSINESS_TIME WITHOUT OVERLAPS)
);
```

Wäre die PRIMARY KEY-Deklaration wie folgt definiert:

```
PRIMARY KEY(product_id, BUSINESS_TIME WITHOUT OVERLAPS)
```

könnte dasselbe Produkt nicht gleichzeitig von zwei verschiedenen Lieferanten geliefert werden.

Einfügen von Daten in eine temporale Tabelle für Anwendungszeitraum

Das Einfügen von Daten in eine temporale Tabelle für Anwendungszeitraum ist mit dem Einfügen von Daten in eine reguläre Tabelle vergleichbar.

Informationen zu diesem Vorgang

Die einzige besondere Anforderung beim Einfügen von Daten in eine temporale Tabelle für Anwendungszeitraum besteht darin, dass die Zeilenbeginn- und -endspalten eingefügt werden müssen. Über diese Spalten wird erfasst, ob eine Zeile aus Sicht der zugeordneten Geschäftsanwendungen gültig ist. Dieser gültige Zeitraum wird als BUSINESS_TIME-Zeitraum bezeichnet. Der Datenbankmanager generiert automatisch eine implizite Prüfbedingung, die sicherstellt, dass der Wert der Beginnspalte des BUSINESS_TIME-Zeitraums kleiner als der Wert der entsprechenden Endspalte ist. Wurde für die Tabelle eine eindeutige Integritätsbedingung oder ein Index mit BUSINESS_TIME WITHOUT OVERLAPS erstellt, müssen Sie sicherstellen, dass sich keine BUSINESS_TIME-Zeiträume überschneiden.

Vorgehensweise

Zum Einfügen von Daten in eine temporale Tabelle für Anwendungszeitraum wird die INSERT-Anweisung verwendet. Beispiel: Die folgenden Daten wurden in die Tabelle eingefügt, die in dem im Abschnitt Erstellen einer temporalen Tabelle für Anwendungszeitraum aufgeführten Beispiel erstellt wurde.

```
INSERT INTO policy_info VALUES('A123',12000,'2008-01-01','2008-07-01');

INSERT INTO policy_info VALUES('A123',16000,'2008-07-01','2009-01-01');

INSERT INTO policy_info VALUES('A123',16000,'2008-06-01','2008-08-01');

INSERT INTO policy_info VALUES('B345',18000,'2008-01-01','2009-01-01');

INSERT INTO policy_info VALUES('C567',20000,'2008-01-01','2009-01-01');
```

Ergebnisse

Es wurden fünf INSERT-Anweisungen ausgegeben, der Tabelle wurden aber nur vier Zeilen hinzugefügt. Die zweite und die dritte INSERT-Anweisung versuchen, Zeilen für die Police A123 hinzuzufügen, aber die zugehörigen BUSINESS_TIME-Zeiträume überschneiden sich. Dadurch ergibt sich Folgendes:

- Die zweite INSERT-Anweisung fügt eine Zeile für `policy_id` A123 mit einem `bus_start`-Wert von 2008-07-01 und einem `bus_end`-Wert von 2009-01-01 ein.
- Die dritte INSERT-Anweisung versucht, eine Zeile für `policy_id` A123 einzufügen. Dieser Versuch schlägt fehl, weil sich der diesbezügliche BUSINESS_TIME-Zeitraum mit dem Zeitraum der vorherigen INSERT-Anweisung überschneidet.

Die Tabelle policy_info wurde mit einem BUSINESS_TIME WITHOUT OVERLAPS-Index erstellt und die dritte INSERT-Anweisung verfügt über einen bus_end-Wert von 2008-08-01, der sich innerhalb des Zeitraums der früheren INSERT-Anweisung befindet.

Die Beginnspalte eines Zeitraums ist inklusiv, während die Endspalte exklusiv ist. Dies bedeutet, dass sich bei der Zeile mit einem bus_end-Wert von 2008-07-01 kein BUSINESS_TIME-Zeitraum mit der Zeile überschneidet, die einen bus_start-Wert von 2008-07-01 enthält. Demzufolge enthält die Tabelle policy_info nun die folgenden Versicherungsdeckungsdaten:

Tabelle 47. Einer temporalen Tabelle für Anwendungszeitraum hinzugefügte Daten (policy_info)

policy_id	coverage	bus_start	bus_end
A123	12000	2008-01-01	2008-07-01
A123	16000	2008-07-01	2009-01-01
B345	18000	2008-01-01	2009-01-01
C567	20000	2008-01-01	2009-01-01

Aktualisieren von Daten in einer temporalen Tabelle für Anwendungszeitraum

Das Aktualisieren einer temporalen Tabelle für Anwendungszeitraum kann dem Aktualisieren von Daten in einer regulären Tabelle entsprechen, die Daten können jedoch auch für bestimmte Zeitpunkte in der Vergangenheit, Gegenwart oder Zukunft aktualisiert werden. Aktualisierungen für bestimmte Zeitpunkte können zu einem automatischen Aufteilen von Zeilen und Einfügen neuer Zeilen in der Tabelle führen.

Informationen zu diesem Vorgang

Neben der regulären UPDATE-Anweisung unterstützen temporale Tabellen für Anwendungszeitraum auch zeitraumbezogene Aktualisierungen, bei denen die Anweisung UPDATE die Klausel FOR PORTION OF BUSINESS_TIME enthält. Eine Zeile ist ein Kandidat für die Aktualisierung, wenn die zugehörige Beginn- und/oder Endspalte für einen Zeitraum in den Zeitraum fällt, der in der Klausel FOR PORTION OF BUSINESS_TIME angegeben ist.

Vorgehensweise

Zum Aktualisieren von Daten in einer temporalen Tabelle für Anwendungszeitraum wird die Anweisung UPDATE verwendet. Beispiel: Es wurde festgestellt, dass die Versicherungsdeckungsdaten für einige Kunden fehlerhaft sind. An der Beispieltabelle vorgenommen, die im Abschnitt „Einfügen von Daten in eine temporale Tabelle für Anwendungszeitraum" aufgeführt ist, werden die folgenden Aktualisierungen vorgenommen.
Die folgende Tabelle enthält die ursprünglichen Daten der Tabelle policy_info.

Tabelle 48. Ursprüngliche Daten in der temporalen Tabelle für Anwendungszeitraum (policy_info)

policy_id	coverage	bus_start	bus_end
A123	12000	2008-01-01	2008-07-01
A123	16000	2008-07-01	2009-01-01
B345	18000	2008-01-01	2009-01-01
C567	20000	2008-01-01	2009-01-01

Die Tabelle policy_info wurde mit einem BUSINESS_TIME WITHOUT OVERLAPS-Index erstellt. Bei Verwendung der regulären UPDATE-Anweisung müssen Sie sicherstellen, dass sich keine BUSINESS_TIME-Zeiträume überschneiden. Die Aktualisierung einer temporalen Tabelle für Anwendungszeitraum unter Verwendung der Klausel FOR PORTION OF BUSINESS_TIME vermeidet Probleme hinsichtlich der Überschneidung von Zeiträumen. Diese Klausel veranlasst Zeilenänderungen und kann dazu führen, dass Zeilen eingefügt werden, wenn der vorhandene Zeitraum für eine zu aktualisierende Zeile nicht vollständig innerhalb des in der Anweisung UPDATE angegebenen Zeitraums liegt.

- Die Versicherungsdeckung für die Police B345 beginnt am 1. März 2008 (2008-03-01) und die Deckung liegt bei 18500:

```
UPDATE policy_info
  SET coverage = 18500, bus_start = '2008-03-01'
  WHERE policy_id = 'B345'
  AND coverage=18000
```

  Die Aktualisierung für die Police B345 verwendet eine reguläre UPDATE-Anweisung. In der Tabelle policy_info ist für policy_id B345 nur eine Zeile vorhanden, sodass es keine potenziellen Überschneidungen von BUSINESS_TIME-Zeiträumen gibt. Als Folge wird der Wert der Spalte bus_start auf 2008-03-01 aktualisiert und der Wert für coverage auf 18500. Dabei ist zu beachten, dass bei Aktualisierungen von Spalten mit einem BUSINESS_TIME-Zeitraum die Klausel FOR PORTION OF BUSINESS_TIME nicht verwendet werden kann.

Tabelle 49. Aktualisierte Police B345

policy_id	coverage	bus_start	bus_end
A123	12000	2008-01-01	2008-07-01
A123	16000	2008-07-01	2009-01-01
B345	18500	2008-03-01	2009-01-01
C567	20000	2008-01-01	2009-01-01

- Die Versicherungsdeckung für die Police C567 für das Jahr 2008 soll 25000 betragen:

```
UPDATE policy_info
   FOR PORTION OF BUSINESS_TIME FROM '2008-01-01' TO '2009-01-01'
   SET coverage = 25000
   WHERE policy_id = 'C567';
```

  Die Aktualisierung der Police C567 gilt für den BUSINESS_TIME-Zeitraum von 2008-01-01 bis 2009-01-01. In der Tabelle policy_info gibt es nur eine Zeile für policy_id C567, die diesen Zeitraum enthält. Der BUSINESS_TIME-Zeitraum liegt innerhalb der Spaltenwerte von bus_start und bus_end für diese Zeile. Als Folge wird der Wert von coverage auf 25000 aktualisiert. Die Spaltenwerte von bus_start und bus_end werden nicht geändert.

Tabelle 50. Aktualisierte Police C567

policy_id	coverage	bus_start	bus_end
A123	12000	2008-01-01	2008-07-01
A123	16000	2008-07-01	2009-01-01
B345	18500	2008-03-01	2009-01-01
C567	25000	2008-01-01	2009-01-01

- Die Versicherungsdeckung für die Police A123 zeigt eine Erhöhung von 12000 auf 16000 am 1. Juli (2008-07-01), aber eine frühere Erhöhung auf 14000 ist nicht vorhanden:

```
UPDATE policy_info
   FOR PORTION OF BUSINESS_TIME FROM '2008-06-01' TO '2008-08-01'
   SET coverage = 14000
   WHERE policy_id = 'A123';
```

  Die Aktualisierung der Police A123 gilt für den BUSINESS_TIME-Zeitraum von 2008-06-01 bis 2008-08-01. In der Tabelle `policy_info` gibt es zwei Zeilen für `policy_id` A123, die einen Teil dieses Zeitraums umfassen.

  1. Der BUSINESS_TIME-Zeitraum ist in der Zeile mit einem `bus_start`-Wert von 2008-01-01 und einem `bus_end`-Wert von 2008-07-01 teilweise enthalten. Diese Zeile überschneidet sich mit dem Beginn des angegebenen Zeitraums, da der früheste Zeitwert im BUSINESS_TIME-Zeitraum höher als der `bus_start`-Wert, aber kleiner als der `bus_end`-Wert der Zeile ist.
  2. Der BUSINESS_TIME-Zeitraum ist in der Zeile mit einem `bus_start`-Wert von 2008-07-01 und einem `bus_end`-Wert von 2009-01-01 teilweise enthalten. Diese Zeile überschneidet sich mit dem Ende des angegebenen Zeitraums, da der späteste Zeitwert im BUSINESS_TIME-Zeitraum höher als der `bus_start`-Wert, aber kleiner als der `bus_end`-Wert der Zeile ist.

  Folglich werden durch die Aktualisierung die folgenden Aktionen ausgelöst:

  1. Wenn sich der Wert von `bus_end` mit dem Beginn des angegebenen Zeitraums überschneidet, wird die Zeile auf den neuen Deckungswert von 14000 aktualisiert. In dieser aktualisierten Zeile ist `bus_start` auf den Wert 2008-06-01 gesetzt, der dem Anfangswert des über die Anweisung UPDATE angegebenen Zeitraums entspricht; der Wert von `bus_end` wird nicht geändert. Mit den ursprünglichen Werten der Zeile wird eine zusätzliche Zeile einfügt, der Wert von `bus_end` wird jedoch auf 2008-06-01 gesetzt. Diese neue Zeile gibt den BUSINESS_TIME-Zeitraum an, in dem der Deckungswert 12000 betrug.
  2. Wenn sich der Wert von `bus_start` mit dem Ende des angegebenen Zeitraums überschneidet, wird die Zeile auf den neuen Deckungswert von 14000 aktualisiert. In dieser aktualisierten Zeile wird der Wert von `bus_start` nicht geändert; `bus_end` wird auf den Wert 2008-08-01 gesetzt, der dem Endwert des über die Anweisung UPDATE angegebenen Zeitraums entspricht. Mit den ursprünglichen Werten der Zeile wird eine zusätzliche Zeile einfügt, der Wert von `bus_start` wird jedoch auf 2008-08-01 gesetzt. Diese neue Zeile gibt den BUSINESS_TIME-Zeitraum an, in dem der Deckungswert 16000 betrug.

Tabelle 51. Aktualisierte Police A123

policy_id	coverage	bus_start	bus_end
A123	12000	2008-01-01	2008-06-01
A123	14000	2008-06-01	2008-07-01
A123	14000	2008-07-01	2008-08-01
A123	16000	2008-08-01	2009-01-01
B345	18500	2008-03-01	2009-01-01
C567	25000	2008-01-01	2009-01-01

Weitere Beispiele

Der folgende Abschnitt enthält weitere Beispiele zur Aktualisierung von temporalen Tabellen für Anwendungszeitraum.

Zusammenführen von Inhalten

Im folgenden Beispiel verwendet eine MERGE-Anweisung die Klausel FOR PORTION OF, um die Tabelle policy_info mit dem Inhalt einer anderen Tabelle (merge_policy) zu aktualisieren.

Tabelle 52. Inhalt der Tabelle 'merge_policy'

policy_id	coverage	bus_start	bus_end
C567	30000	2008-10-01	2010-05-01
H789	16000	2008-10-01	2010-05-01

1. Erstellen Sie globale Variablen für die Datumsangaben FROM und TO der Klausel FOR PORTION OF.

```
CREATE VARIABLE sdate DATE default '2008-10-01';
CREATE VARIABLE edate DATE default '2010-05-01';
```

2. Führen Sie eine MERGE-Anweisung aus, die den Inhalt der Tabelle merge_policy mit dem Inhalt der Tabelle policy_info zusammenführt, die aus den Aktualisierungen im Abschnitt „Vorgehensweise" weiter oben resultiert ist.

```
MERGE INTO policy_info pi1
   USING (SELECT policy_id, coverage, bus_start, bus_end FROM merge_policy) mp2
   ON (pi1.policy_id = mp2.policy_id)
WHEN MATCHED THEN
     UPDATE FOR PORTION OF BUSINESS_TIME FROM sdate TO edate
        SET pi1_coverage = mp2.coverage
WHEN NOT MATCHED THEN
      INSERT (policy_id, coverage, bus_start, bus_end)
         VALUES (mp2.policy_id, mp2.coverage, mp2.bus_start, mp2.bus_end)
```

Die policy_id C567 ist für beide Tabellen identisch. Der bus_start-Wert von C567 in merge_policy überschneidet sich mit dem bus_end-Wert von C567 in policy_info. Diese Anweisung resultiert in den folgenden Aktionen:

- Der bus_end-Wert für die Deckung von 25000 wird auf 2008-10-01 gesetzt.
- Für die Deckung von 30000 mit den bus_start- und bus_end-Werten aus merge_policy wird eine neue Zeile eingefügt.

Die policy_id H789 ist nur in merge_policy vorhanden, daher wird in policy_info eine neue Zeile eingefügt.

Tabelle 53. Zusammengeführte aktualisierte Daten in einer temporalen Tabelle für Anwendungszeitraum (policy_info)

policy_id	coverage	bus_start	bus_end
A123	12000	2008-01-01	2008-06-01
A123	14000	2008-06-01	2008-07-01
A123	14000	2008-07-01	2008-08-01
A123	16000	2008-08-01	2009-01-01
B345	18000	2008-03-01	2009-01-01
C567	25000	2008-01-01	2008-10-01
C567	30000	2008-10-01	2010-05-01
H789	16000	2008-10-01	2010-05-01

Aktualisieren von Zielen

Die Klausel FOR PORTION OF BUSINESS_TIME kann nur dann verwen-

det werden, wenn das Ziel der UPDATE-Anweisung eine Tabelle oder eine Sicht ist. Die folgenden Aktualisierungen geben Fehler zurück.

```
UPDATE (SELECT * FROM policy_info) FOR PORTION OF BUSINESS_TIME
   FROM '2008-01-01' TO '06-15-2008' SET policy_id = policy_id + 1;

UPDATE (SELECT * FROM policy_info FOR BUSINESS_TIME AS OF '2008-01-01')
   FOR PORTION OF BUSINESS_TIME FROM '2008-01-01' TO '06-15-2008'
   SET policy_id = policy_id + 1;
```

Aktualisieren einer Sicht

Eine Sicht mit Verweisen auf eine temporale Tabelle für Anwendungszeitraum ist aktualisierbar. Mit der folgenden UPDATE-Anweisung wird die Tabelle `policy_info` aktualisiert.

```
CREATE VIEW viewC AS SELECT * FROM policy_info;
UPDATE viewC SET coverage = coverage + 5000;
```

Eine Sicht mit einer temporalen Tabelle für Anwendungszeitraum in der zugehörigen FROM-Klausel, die eine Zeitraumangabe enthält, ist ebenfalls aktualisierbar. Diese Bedingung unterscheidet sich von den Sichten in temporalen Tabellen für Systemzeitraum und bitemporalen Tabellen.

```
CREATE VIEW viewD AS SELECT * FROM policy_info
   FOR BUSINESS_TIME AS OF CURRENT DATE;
UPDATE viewD SET coverage = coverage - 1000;
```

Für Sichten mit Verweisen auf temporale Tabellen für Anwendungszeitraum oder bitemporale Tabellen kann eine FOR PORTION OF-Aktualisierungsklausel eingefügt werden. Aktualisierungen dieser Art werden an die temporalen Tabellen weitergegeben, auf die in der FROM-Klausel der Sichtdefinition verwiesen wird.

```
CREATE VIEW viewE AS SELECT * FROM policy_info;
UPDATE viewE FOR PORTION OF BUSINESS_TIME
   FROM '2009-01-01' TO '2009-06-01' SET coverage = coverage + 500;
```

Löschen von Daten aus einer temporalen Tabelle für Anwendungszeitraum

Beim Löschen von Daten aus einer temporalen Tabelle für Anwendungszeitraum werden Zeilen in der Tabelle gelöscht. Dies kann dazu führen, dass neue Zeilen in die temporale Tabelle für Anwendungszeitraum eingefügt werden.

Informationen zu diesem Vorgang

Neben der regulären DELETE-Anweisung unterstützen temporale Tabellen für Anwendungszeitraum auch zeitraumbezogene Aktualisierungen, bei denen die DELETE-Anweisung die Klausel FOR PORTION OF BUSINESS_TIME enthält. Eine Zeile ist ein Kandidat für einen Löschvorgang, wenn die zugehörige Beginn- und/oder Endspalte für einen Zeitraum in den Zeitraum fällt, der in der Klausel FOR PORTION OF BUSINESS_TIME angegeben ist.

Vorgehensweise

Zum Löschen von Daten aus einer temporalen Tabelle für Anwendungszeitraum wird die Anweisung DELETE FROM verwendet. Es wurde beispielsweise festgestellt, dass die Police A123 zwischen dem 15. Juni 2008 und dem 15. August 2008 nicht über eine Deckung verfügen dürfte. Die betreffenden Daten sollten deshalb in der im Abschnitt Aktualisieren von Daten in einer temporalen Tabelle für Anwendungszeitraum aktualisierten Tabelle gelöscht werden.

```
DELETE FROM policy_info
   FOR PORTION OF BUSINESS_TIME FROM '2008-06-15' TO '2008-08-15'
   WHERE policy_id = 'A123';
```

Ergebnisse

Die ursprünglichen Daten aus der Tabelle policy_info lauten wie folgt:

Tabelle 54. Daten in der temporalen Tabelle für Anwendungszeitraum (policy_info) vor der DELETE-Anweisung

policy_id	coverage	bus_start	bus_end
A123	12000	2008-01-01	2008-06-01
A123	14000	2008-06-01	2008-07-01
A123	14000	2008-07-01	2008-08-01
A123	16000	2008-08-01	2009-01-01
B345	18000	2008-03-01	2009-01-01
C567	25000	2008-01-01	2009-01-01

Das Löschen von Daten aus einer temporalen Tabelle für Anwendungszeitraum unter Verwendung der Klausel FOR PORTION OF BUSINESS_TIME führt dazu, dass Zeilen gelöscht werden und möglicherweise Zeilen eingefügt werden, wenn der Zeitraum für eine Zeile einen Teil des in der Anweisung DELETE FROM angegebenen Zeitraums abdeckt. Das Löschen von Daten im Zusammenhang mit der Police A123 bezieht sich auf den BUSINESS_TIME-Zeitraum vom 15.06.2008 (2008-06-15) bis zum 15.08.2008 (2008-08-15). In der Tabelle policy_info befinden sich drei Zeilen für policy_id A123, die diesen Zeitraum vollständig oder teilweise abdecken.

Die Aktualisierung der Police A123 hat Auswirkungen auf die temporale Tabelle für Systemzeitraum sowie die zugehörige Protokolltabelle und führt dazu, dass die folgenden Aktionen ausgeführt werden:

- Es gibt eine Zeile, für die der BUSINESS_TIME-Zeitraum in der Anweisung DELETE FROM den gesamten Zeitraum für eine Zeile abdeckt. Die Zeile mit einem bus_start-Wert von 2008-07-01 und einem bus_end-Wert von 2008-08-01 wird gelöscht.
- Wenn nur der bus_end-Wert innerhalb des angegebenen Zeitraums liegt, wird die Zeile gelöscht. Eine neue Zeile mit den ursprünglichen Werten aus der gelöschten Zeile wird eingefügt, wobei der bus_end-Wert allerdings auf 2008-06-15 gesetzt ist.
- Wenn nur der bus_start-Wert innerhalb des angegebenen Zeitraums liegt, wird die Zeile gelöscht. Eine neue Zeile mit den ursprünglichen Werten aus der gelöschten Zeile wird eingefügt, wobei der bus_start-Wert allerdings auf 2008-08-15 gesetzt ist.

Tabelle 55. Daten in der temporalen Tabelle für Anwendungszeitraum (policy_info) nach der DELETE-Anweisung

policy_id	coverage	bus_start	bus_end
A123	12000	2008-01-01	2008-06-01
A123	14000	2008-06-01	2008-06-15
A123	16000	2008-08-15	2009-01-01
B345	18000	2008-03-01	2009-01-01
C567	25000	2008-01-01	2009-01-01

Beispiel

Der folgende Abschnitt enthält weitere Beispiele zum Löschen von temporalen Tabellen für Anwendungszeitraum.

Löschen von Zielen

Die Klausel FOR PORTION OF BUSINESS_TIME kann nur dann verwendet werden, wenn das Ziel der DELETE-Anweisung eine Tabelle oder eine Sicht ist. Die folgende DELETE-Anweisung gibt einen Fehler zurück:

```
DELETE FROM (SELECT * FROM policy_info) FOR PORTION OF BUSINESS_TIME
   FROM '2008-01-01' TO '2008-06-15';
```

Abfragen von Daten in einer temporalen Tabelle für Anwendungszeitraum

Durch das Abfragen einer temporalen Tabelle für Anwendungszeitraum können Ergebnisse für einen bestimmten Zeitraum zurückgegeben werden.

Informationen zu diesem Vorgang

Beim Abfragen einer temporalen Tabelle für Anwendungszeitraum kann FOR BUSINESS_TIME in die FROM-Klausel eingefügt werden. Mithilfe von FOR BUSINESS_TIME-Angaben kann der aktuelle, der frühere und der zukünftige Status der Daten abgefragt werden. Die Zeiträume werden wie folgt angegeben:

AS OF *wert1*

Enthält alle Zeilen, bei denen der Anfangswert für den Zeitraum kleiner-gleich *wert1* und der Endwert für den Zeitraum größer als *wert1* ist.

FROM *wert1* **TO** *wert2*

Enthält alle Zeilen, bei denen der Anfangswert für den Zeitraum größer-gleich *wert1* und der Endwert für den Zeitraum kleiner als *wert2* ist.Dies bedeutet, dass die Anfangszeit in den Zeitraum eingeschlossen ist, die Endzeit jedoch nicht.

BETWEEN *wert1* **AND** *wert2*

Enthält alle Zeilen, bei denen sich ein beliebiger Zeitraum mit einem beliebigen Zeitpunkt zwischen *wert1* und *wert2* überschneidet. Eine Zeile wird zurückgegeben, wenn der Anfangswert für den Zeitraum kleiner-gleich *wert2* und der Endwert für den Zeitraum größer als *wert1* ist.

Im folgenden Abschnitt sind einige Beispielabfragen aufgelistet.

Vorgehensweise

Zum Abfragen einer temporalen Tabelle für Anwendungszeitraum wird die Anweisung SELECT verwendet. Beispielsweise fordert jede der folgenden Abfragen Policeninformationen für `policy_id` A123 aus der im Abschnitt „ Aktualisieren von Daten in einer temporalen Tabelle für Anwendungszeitraum” aufgeführten Ergebnistabelle an. Jede Abfrage verwendet eine Variante der Zeitraumangabe.
Die Tabelle `policy_info` sieht wie folgt aus:

Tabelle 56. Temporale Tabelle für Anwendungszeitraum: policy_info

policy_id	coverage	bus_start	bus_end
A123	12000	2008-01-01	2008-06-01
A123	14000	2008-06-01	2008-06-15
A123	16000	2008-08-15	2009-01-01

Tabelle 56. Temporale Tabelle für Anwendungszeitraum: policy_info (Forts.)

policy_id	coverage	bus_start	bus_end
B345	18000	2008-03-01	2009-01-01
C567	25000	2008-01-01	2009-01-01

- Abfrage ohne Zeitraumangabe. Beispiel:

```
SELECT policy_id, coverage, bus_start, bus_end
   FROM policy_info
   where policy_id = 'A123'
```

  Diese Abfrage gibt alle drei Zeilen für die Police A123 zurück.

```
A123, 12000, 2008-01-01, 2008-06-01
A123, 14000, 2008-06-01, 2008-06-15
A123, 16000, 2008-08-15, 2009-01-01
```

- Abfrage mit Angabe von FOR BUSINESS_TIME AS OF. Beispiel:

```
SELECT policy_id, coverage, bus_start, bus_end
   FROM policy_info
   FOR BUSINESS_TIME AS OF '2008-07-15'
   where policy_id = 'A123'
```

  Diese Abfrage gibt keine Zeilen zurück. Es gibt keine Zeilen für A123, bei denen der Anfangswert für den Zeitraum kleiner-gleich *2008-07-15* und der Endwert für den Zeitraum größer als *2008-07-15* ist. Die Police A123 verfügte am 15.07.2008 (*2008-07-15*) über keine Deckung.

- Abfrage mit Angabe von FOR BUSINESS_TIME FROM...TO. Beispiel:

```
SELECT policy_id, coverage, bus_start, bus_end
   FROM policy_info
   FOR BUSINESS_TIME FROM
      '2008-01-01' TO '2008-06-15'
   where policy_id = 'A123'
```

  Diese Abfrage gibt zwei Zeilen zurück. Die Beginnspalte eines Zeitraums ist inklusiv, die Endspalte ist exklusiv. Die Zeile mit einem 'bus_end'-Wert von 2008-06-15 ist bis zum 14.06.2008 um Mitternacht gültig und liegt somit vor *wert2*.

```
A123, 12000, 2008-01-01, 2008-06-01
A123, 14000, 2008-06-01, 2008-06-15
```

- Abfrage mit Angabe von FOR BUSINESS_TIME BETWEEN...AND. Beispiel:

```
SELECT policy_id, coverage, bus_start, bus_end
   FROM policy_info
   FOR BUSINESS_TIME BETWEEN
      '0001-01-01' AND '2008-01-01'
```

  Diese Abfrage gibt zwei Zeilen zurück. Die Zeilen mit einem 'bus_start'-Wert von 2008-01-01 entsprechen *wert1* und werden zurückgegeben, weil die Zeit für den Anfang eines Zeitraums eingeschlossen ist. Dabei ist zu beachten, dass eine Zeile, die einen 'bus_end'-Spaltenwert von 2008-01-01 aufweist, zurückgegeben wird, da die Endzeit *wert1* entspricht und die Endzeit eines Zeitraums eingeschlossen ist.

```
A123, 12000, 2008-01-01, 2008-06-01
C567, 25000, 2008-01-01, 2009-01-01
```

Weitere Beispiele

Der folgende Abschnitt enthält weitere Beispiele zur Abfrage von temporalen Tabellen für Anwendungszeitraum.

Abfragen einer Sicht

Eine Sicht kann wie eine temporale Tabelle für Anwendungszeitraum abgefragt werden. Zeitraumangaben (FOR BUSINESS_TIME) können nach dem Sichtverweis angegeben werden.

```
CREATE VIEW policy_year_end(policy, amount, start_date, end_date)
   AS SELECT * FROM policy_info;

SELECT * FROM policy_year_end FOR BUSINESS_TIME AS OF '2008-12-31';
```

Die Anweisung SELECT für die Sicht `policy_year_end` führt eine Abfrage für die Tabelle `policy_info` aus und gibt alle Policen zurück, die Ende 2008 gültig waren.

```
A123, 16000, 2008-08-15, 2009-01-01
B345, 18000, 2008-03-01, 2009-01-01
C567, 25000, 2008-01-01, 2009-01-01
```

Wenn eine Sichtdefinition eine Zeitraumangabe enthält, können Abfragen für diese Sicht keine Zeitraumangaben enthalten. Die folgenden Anweisungen geben einen Fehler zurück, der auf mehrere Zeitraumangaben zurückzuführen ist:

```
CREATE VIEW all_policies AS SELECT * FROM policy_info;
   FOR BUSINESS_TIME AS OF '2008-02-28';

SELECT * FROM all_policies FOR BUSINESS_TIME BETWEEN
   FOR BUSINESS_TIME AS OF '2008-10-01';
```

Anwendungslaufzeit für Sitzung festlegen

Durch Festlegen der Anwendungszeit im Sonderregister CURRENT TEMPORAL BUSINESS_TIME werden die Änderungen, die bei der Ausführung einer Anwendung für bestimmte Zeitpunkte erforderlich sind, möglicherweise teilweise oder ganz überflüssig.

Informationen zu diesem Vorgang

Wenn Sie eine Ihrer Anwendungen für eine temporale Tabelle für Anwendungszeitraum ausführen wollen, um den Status Ihres Unternehmens zu einer Reihe verschiedener Daten abzufragen, können Sie das Datum in einem Sonderregister festlegen. Wenn Sie Ihre Daten mit Stand von heute oder Ende des letzten Quartals abfragen müssen oder künftige Ereignisse wie den Stand eines Datums in der Zukunft simulieren, ist es unter Umständen nicht möglich, die Anwendung zu ändern und AS OF-Spezifikationen zu den einzelnen SQL-Anweisungen hinzuzufügen. Diese Beschränkung ist bei der Verwendung von Paketanwendungen wahrscheinlich. Sie können bei solchen Szenarios das Datum mit dem Sonderregister CURRENT TEMPORAL BUSINESS_TIME auf Sitzungsebene festlegen.

Die Einstellung des Sonderregisters CURRENT TEMPORAL BUSINESS_TIME hat keine Auswirkungen auf reguläre Tabellen. Die im Sonderregister festgelegte Zeitmarke wird nur von Abfragen für temporale Tabellen mit einem aktivierten Zeitraum BUSINESS_TIME (temporale Tabellen für Anwendungszeitraum und bitemporale Tabellen) verwendet. Es gibt auch keine Auswirkungen auf DDL-Anweisungen.

Anmerkung: Wenn das Sonderregister CURRENT TEMPORAL BUSINESS_TIME auf einen Nichtnullwert eingestellt ist, werden Datenänderungsanweisungen wie INSERT, UPDATE, DELETE und MERGE für temporale Tabellen für Anwendungszeitraum unterstützt. Dieses Verhalten unterscheidet sich vom Sonderregister CUR-

RENT TEMPORAL SYSTEM_TIME, das Datenänderungsanweisungen für temporale Tabellen für Systemzeitraum und bitemporale Tabellen blockiert.

Die Einstellung für die Bindeoption BUSTIMESENSITIVE bestimmt, ob Verweise auf temporale Tabellen für Anwendungszeitraum und bitemporale Tabellen in statischen und dynamischen SQL-Anweisungen in einem Paket durch den Wert des Sonderregisters CURRENT TEMPORAL BUSINESS_TIME betroffen sind. Die Bindeoption kann mit dem Wert YES oder NO definiert werden. Verwenden Sie bei SQL-Prozeduren die Prozedur SET_ROUTINE_OPTS zum Definieren der mit dem Binden verwandten Optionen, der sogenannten Abfragecompilervariablen.

Vorgehensweise

Wenn dieses Sonderregister auf einen Nichtnullwert eingestellt wird, geben Anwendungen, die eine Abfrage absetzen, Daten zum Stand des betreffenden Datums zurück. In den folgenden Beispielen werden Informationen aus den im Abschnitt „ Löschen von Daten aus einer temporalen Tabelle für Anwendungszeitraum ” aufgeführten Ergebnistabellen abgefragt.

- Legen Sie für das Sonderregister einen Nichtnullwert fest und fragen Sie Daten zum Stand dieses Datums ab. Beispiel:

```
SET CURRENT TEMPORAL BUSINESS_TIME = '2008-01-01';
SELECT * FROM policy_info;
```

- Legen Sie für das Sonderregister eine Zeitmarke fest und verweisen Sie in den Sichtdefinitionen auf eine temporale Tabelle für Anwendungszeitraum.

```
CREATE VIEW view1 AS SELECT policy_id, coverage FROM policy_info;
CREATE VIEW view2 AS SELECT * FROM regular_table
   WHERE col1 IN (SELECT coverage FROM policy_info);
SET CURRENT TEMPORAL BUSINESS_TIME = '2008-01-01';
SELECT * FROM view1;
SELECT * FROM view2;
```

- Legen Sie für das Sonderregister ein Datum in der Vergangenheit fest und setzen Sie eine Abfrage ab, die eine Zeitraumspezifikation enthält. Beispiel:

```
SET CURRENT TEMPORAL BUSINESS_TIME = CURRENT DATE – 1 YEAR;
SELECT * FROM policy_info FOR BUSINESS_TIME AS OF '2008-01-01';
```

Ergebnisse

Die Tabelle `policy_info` sieht wie folgt aus:

Tabelle 57. Daten in der temporalen Tabelle für Anwendungszeitraum (policy_info) nach der DELETE-Anweisung

policy_id	coverage	bus_start	bus_end
A123	12000	2008-01-01	2008-06-01
A123	14000	2008-06-01	2008-06-15
A123	16000	2008-08-15	2009-01-01
B345	18000	2008-03-01	2009-01-01
C567	25000	2008-01-01	2009-01-01

- Bei der Anforderung von Daten mit Stand vom 2008-01-01 wird die Tabelle `policy_info` abgefragt. Die Abfrage wird implizit wie folgt umgeschrieben:

```
SELECT * FROM policy_info FOR BUSINESS_TIME AS OF '2008-01-01';
```

  Die Abfrage gibt Folgendes zurück:

```
A123, 12000, 2008-01-01, 2008-06-01
C567, 25000, 2008-01-01, 2009-01-01
```

- Die Abfrage für view1 wird implizit wie folgt umgeschrieben:

  ```
  SELECT * FROM view1 FOR BUSINESS_TIME AS OF CURRENT TEMPORAL BUSINESS_TIME;
  ```

  Danach wie folgt:

  ```
  SELECT policy_id, coverage FROM policy_info
     FOR BUSINESS_TIME AS OF '2008-01-01';
  ```

  Die Abfrage gibt Folgendes zurück:

  ```
  A123, 12000
  C567, 25000
  ```

  Die Abfrage für view2 umfasst eine Sicht für eine reguläre Tabelle, die auf eine temporale Tabelle für Anwendungszeitraum verweist. Dies bewirkt eine implizite Beziehung zwischen einer regulären Tabelle und dem Sonderregister. Die Abfrage wird implizit wie folgt umgeschrieben:

  ```
  SELECT * FROM view2 FOR BUSINESS_TIME AS OF CURRENT TEMPORAL BUSINESS_TIME;
  ```

  Danach wie folgt:

  ```
  SELECT * FROM regular_table WHERE col1 in (SELECT coverage FROM policy_info
     FOR BUSINESS_TIME AS OF '2008-01-01');
  ```

  Die Anweisung SELECT gibt Zeilen zurück, wobei die Werte für col1 mit den Werten für coverage übereinstimmen.
- Da mehrere Zeitraumspezifikationen vorhanden sind, wird ein Fehler zurückgegeben. Für das Sonderregister wurde ein Nichtnullwert festgelegt und auch in der Abfrage wurde eine Zeit angegeben.

Bitemporale Tabellen

Eine bitemporale Tabelle kombiniert die Langzeitdatenprotokollierung einer temporalen Tabelle für Systemzeitraum mit den zeitpunktbasierten Datenspeicherungsfunktionen einer temporalen Tabelle für Anwendungszeitraum. Mit bitemporalen Tabellen können Sie sowohl benutzerbasierte Zeitrauminformationen als auch systembasierte Protokollinformationen verwalten.

Bitemporale Tabellen verhalten sich wie eine Kombination aus temporalen Tabellen für Systemzeitraum und temporalen Tabellen für Anwendungszeitraum. Alle geltenden Einschränkungen für temporale Tabellen für Systemzeitraum und temporale Tabellen für Anwendungszeitraum gelten auch für bitemporale Tabellen.

Erstellen einer bitemporalen Tabelle

Mit der Erstellung einer bitemporalen Tabelle kann die Funktion zur Protokollierung von Daten einer temporalen Tabelle für Systemzeitraum mit den zeitspezifischen Datenspeicherfunktionen einer temporalen Tabelle für Anwendungszeitraum kombiniert werden.

Informationen zu diesem Vorgang

Bei der Erstellung einer bitemporalen Tabelle werden die Schritte zur Erstellung einer temporalen Tabelle für Systemzeitraum mit den Schritten zur Erstellung einer temporalen Tabelle für Anwendungszeitraum kombiniert.

- Fügen Sie in die Anweisung CREATE TABLE einen SYSTEM_TIME-Zeitraum und einen BUSINESS_TIME-Zeitraum ein.
- Erstellen Sie eine Protokolltabelle, in die alte Zeilen aus der bitemporalen Tabelle eingefügt werden.

- Fügen Sie die Versionssteuerung hinzu (ADD VERSIONING), um eine Verbindung zwischen der bitemporalen Tabelle und der Protokolltabelle herzustellen.
- Optional kann festgelegt werden, dass sich überschneidende Zeiträume von BUSINESS_TIME nicht zulässig sind und die Werte in Bezug auf alle Zeiträume eindeutig sein sollen.

Die Beispiele im folgenden Abschnitt zeigen die Erstellung einer Tabelle, in der Informationen zu Versicherungspolicen für die Kunden einer Versicherungsgesellschaft gespeichert werden.

Vorgehensweise

Gehen Sie wie folgt vor, um eine bitemporale Tabelle zu erstellen:

1. Erstellen Sie eine Tabelle mit dem Attribut SYSTEM_TIME und dem Attribut BUSINESS_TIME. Beispiel:

```
CREATE TABLE policy_info
(
 policy_id       CHAR(4) NOT NULL,
 coverage        INT NOT NULL,
 bus_start       DATE NOT NULL,
 bus_end         DATE NOT NULL,
 sys_start   TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW BEGIN,
 sys_end  TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW END,
 ts_id   TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS TRANSACTION START ID,
 PERIOD BUSINESS_TIME (bus_start, bus_end),
 PERIOD SYSTEM_TIME (sys_start, sys_end)
) in policy_space;
```

2. Erstellen Sie eine Protokolltabelle. Beispiel:

```
CREATE TABLE hist_policy_info
(
 policy_id       CHAR(4) NOT NULL,
 coverage        INT NOT NULL,
 bus_start       DATE NOT NULL,
 bus_end         DATE NOT NULL,
 sys_start  TIMESTAMP(12) NOT NULL,
 sys_end  TIMESTAMP(12) NOT NULL,
 ts_id    TIMESTAMP(12)
) in hist_space;
```

   Sie können auch eine Protokolltabelle mit denselben Namen und Beschreibungen wie für die Spalten der temporalen Tabelle für Systemzeitraum erstellen. Verwenden Sie dazu die Klausel LIKE der Anweisung CREATE TABLE. Beispiel:

```
CREATE TABLE hist_policy_info LIKE policy_info in hist_space;
```

3. Fügen Sie der bitemporalen Tabelle die Versionssteuerung hinzu. Beispiel:

```
ALTER TABLE policy_info ADD VERSIONING USE HISTORY TABLE hist_policy_info;
```

4. Optional: Erstellen Sie einen eindeutigen Index, der den BUSINESS_TIME-Zeitraum enthält. Beispiel:

```
CREATE UNIQUE INDEX ix_policy
   ON policy_info (policy_id, BUSINESS_TIME WITHOUT OVERLAPS);
```

Ergebnisse

In der Tabelle `policy_info` wird der Versicherungsdeckungswert für einen Kunden gespeichert. Die auf den Zeitraum BUSINESS_TIME bezogenen Spalten (`bus_start` und `bus_end`) geben an, ob der Wert für die Versicherungsdeckung gültig ist.Die auf den Zeitraum SYSTEM_TIME bezogenen Spalten (`sys_start` und `sys_end`) zei-

gen an, ob eine Zeile für den Deckungswert aktuell ist. Die Spalte ts_id gibt die Startzeit der Ausführung einer Transaktion an, die Auswirkungen auf die Zeile hat.

Tabelle 58. Erstellte bitemporale Tabelle (policy_info)

policy_id	coverage	bus_start	bus_end	sys_start	sys_end	ts_id

Die Protokolltabelle hist_policy_info erhält die alten Zeilen aus der Tabelle policy_info.

Tabelle 59. Erstellte Protokolltabelle (hist_policy_info)

policy_id	coverage	bus_start	bus_end	sys_start	sys_end	ts_id

Der Index ix_policy mit BUSINESS_TIME WITHOUT OVERLAPS als letzter Spalte in der Spaltenliste für Indexschlüssel stellt sicher, dass für die Versicherungsdeckungswerte der Kunden keine sich überschneidenden Zeiträume vorhanden sind.

Beispiel

Der folgende Abschnitt enthält weitere Beispiele zur Erstellung von bitemporalen Tabellen.

Verdecken von Spalten

Das folgende Beispiel erstellt die Tabelle policy_info, wobei die TIMESTAMP(12)-Spalten (sys_start, sys_end und ts_id) als implizit verdeckt markiert sind.

```
CREATE TABLE policy_info
(
 policy_id       CHAR(4) NOT NULL,
 coverage        INT NOT NULL,
 bus_start       DATE NOT NULL,
 bus_end         DATE NOT NULL,
 sys_start   TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW BEGIN IMPLICITLY HIDDEN,
 sys_end  TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW END IMPLICITLY HIDDEN,
 ts_id   TIMESTAMP(12) GENERATED ALWAYS AS TRANSACTION START ID IMPLICITLY HIDDEN,
 PERIOD BUSINESS_TIME (bus_start, bus_end),
 PERIOD SYSTEM_TIME (sys_start, sys_end)
) in policy_space;
```

Wenn die Protokolltabelle hist_policy_info mit der Klausel LIKE der Anweisung CREATE TABLE erstellt wird, übernimmt die Protokolltabelle das Attribut IMPLICITLY HIDDEN von der Tabelle policy_info.

Einfügen von Daten in eine bitemporale Tabelle

Das Einfügen von Daten in eine bitemporale Tabelle ist mit dem Einfügen von Daten in eine temporale Tabelle für Anwendungszeitraum vergleichbar.

Informationen zu diesem Vorgang

Geben Sie beim Einfügen von Daten in eine bitemporale Tabelle Beginn- und Endspalten an, die den Zeitraum erfassen, zu dem eine Zeile aus Sicht der zugeordneten Geschäftsanwendungen gültig ist. Dieser gültige Zeitraum wird als BUSINESS_TIME-Zeitraum bezeichnet. Der Datenbankmanager generiert automatisch eine implizite Prüfbedingung, die sicherstellt, dass der Wert der Beginnspalte des BUSINESS_TIME-Zeitraums kleiner als der Wert der entsprechenden Endspalte ist. Wur-

de für die Tabelle eine eindeutige Integritätsbedingung oder ein Index mit BUSINESS_TIME WITHOUT OVERLAPS erstellt, stellt dies sicher, dass sich keine BUSINESS_TIME-Zeiträume überschneiden.

Vorgehensweise

Zum Einfügen von Daten in eine bitemporale Tabelle wird die INSERT-Anweisung verwendet. Beispiel: Die folgenden Daten wurden am 31. Januar 2010 (2010-01-31) in die Tabelle eingefügt, die in dem im Abschnitt „ Erstellen einer bitemporalen Tabelle" aufgeführten Beispiel erstellt wurde.

```
INSERT INTO policy_info(policy_id, coverage, bus_start, bus_end)
   VALUES('A123',12000,'2008-01-01','2008-07-01');

INSERT INTO policy_info(policy_id, coverage, bus_start, bus_end)
   VALUES('A123',16000,'2008-07-01','2009-01-01');

INSERT INTO policy_info(policy_id, coverage, bus_start, bus_end)
   VALUES('B345',18000,'2008-01-01','2009-01-01');

INSERT INTO policy_info(policy_id, coverage, bus_start, bus_end)
   VALUES('C567',20000,'2008-01-01','2009-01-01');
```

Ergebnisse

Die Tabelle `policy_info` enthält nun die folgenden Versicherungsdeckungsdaten. Die Einträge der Spalten `sys_start`, `sys_end` und `ts_id` werden vom Datenbankmanager generiert. Die Beginnspalte eines Zeitraums ist inklusiv, während die Endspalte exklusiv ist. Dies bedeutet, dass sich bei der Zeile mit einem `bus_end`-Wert von 2008-07-01 kein BUSINESS_TIME-Zeitraum mit der Zeile überschneidet, die einen `bus_start`-Wert von 2008-07-01 enthält.

Tabelle 60. Zu einer bitemporalen Tabelle (policy_info) hinzugefügte Daten

policy_id	coverage	bus_start	bus_end	sys_start	sys_end	ts_id
A123	12000	2008-01-01	2008-07-01	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
A123	16000	2008-07-01	2009-01-01	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
B345	18000	2008-01-01	2009-01-01	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
C567	20000	2008-01-01	2009-01-01	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000

Die Protokolltabelle `hist_policy_info` bleibt leer, weil bei einer Einfügung keine Protokollzeilen generiert werden.

Tabelle 61. Protokolltabelle (hist_policy_info) nach Einfügung

policy_id	coverage	bus_start	bus_end	sys_start	sys_end	ts_id

Aktualisieren von Daten in bitemporaler Tabelle

Das Aktualisieren von Daten in einer bitemporalen Tabelle führt dazu, dass Zeilen zu der zugehörigen Protokolltabelle und potenziell Zeilen zu der bitemporalen Tabelle hinzugefügt werden.

Informationen zu diesem Vorgang

Neben der regulären UPDATE-Anweisung unterstützen bitemporale Tabellen auch zeitraumbezogene Aktualisierungen, bei denen die Anweisung UPDATE die Klausel FOR PORTION OF BUSINESS_TIME enthält. Eine Zeile ist ein Kandidat für die Aktualisierung, wenn die zugehörige Spalte für den Beginn des Zeitraums und/oder die Spalte für das Ende des Zeitraums unter den Zeitraum fällt, der in der Klausel FOR PORTION OF BUSINESS_TIME angegeben ist. Alle vorhandenen betroffenen Zeilen werden in die Protokolltabelle kopiert, bevor sie aktualisiert werden.

Vorgehensweise

Zum Aktualisieren von Daten in einer bitemporalen Tabelle wird die Anweisung UPDATE verwendet, um Datenzeilen zu ändern. Beispiel: Es wurde festgestellt, dass die Versicherungsdeckungswerte für zwei Kunden fehlerhaft sind. Die folgenden Daten wurden am 28. Februar 2011 (2011-02-28) in der Beispieltabelle aktualisiert, in die Daten eingefügt wurden (siehe Abschnitt „Einfügen von Daten in eine bitemporale Tabelle").
Die folgende Tabelle enthält die ursprünglichen Daten der Tabelle `policy_info`.

Tabelle 62. Ursprüngliche Daten in der bitemporalen Tabelle (policy_info)

policy_id	coverage	bus_start	bus_end	sys_start	sys_end	ts_id
A123	12000	2008-01-01	2008-07-01	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
A123	16000	2008-07-01	2009-01-01	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
B345	18000	2008-01-01	2009-01-01	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
C567	20000	2008-01-01	2009-01-01	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000

Beim Aktualisieren einer bitemporalen Tabelle mit der Klausel PORTION OF BUSINESS_TIME werden Zeilen geändert. Das kann dazu führen, dass Zeilen eingefügt werden, wenn der vorhandene Zeitraum für zu aktualisierende Zeilen nicht vollständig innerhalb des in der Anweisung UPDATE angegebenen Zeitraums liegt.

- Die Versicherungsdeckung für die Police B345 beginnt am 1. März 2008 (2008-03-01):

```
UPDATE policy_info
  SET bus_start='2008-03-01'
  WHERE policy_id = 'B345'
  AND coverage = 18000;
```

  Die Aktualisierung für die Police B345 verwendet eine reguläre UPDATE-Anweisung. In der Tabelle `policy_info` ist für `policy_id` B345 nur eine Zeile vorhanden, sodass es keine potenziellen Überschneidungen von BUSINESS_TIME-Zeiträumen gibt. Dadurch passiert Folgendes:

1. Der Wert der Spalte bus_start wird auf 2008-03-01 aktualisiert. Dabei ist zu beachten, dass bei Aktualisierungen von Spalten mit einem BUSINESS_TIME-Zeitraum die Klausel FOR PORTION OF BUSINESS_TIME nicht verwendet werden kann.
2. Der Datenbankmanager aktualisiert die Werte von sys_start und ts_id auf das Datum der Aktualisierung.
3. Die ursprüngliche Zeile wird in die Protokolltabelle versetzt. Der Datenbankmanager aktualisiert den Wert für sys_end auf das Datum der Aktualisierung.

Die folgenden Tabellen geben Aufschluss über die Aktualisierung für die Police B345.

Tabelle 63. Bitemporale Tabelle (policy_info) nach Aktualisierung von Police B345

policy_id	coverage	bus_start	bus_end	sys_start	sys_end	ts_id
A123	12000	2008-01-01	2008-07-01	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
A123	16000	2008-07-01	2009-01-01	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
B345	18000	2008-03-01	2009-01-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000
C567	20000	2008-01-01	2009-01-01	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000

Tabelle 64. Protokolltabelle (hist_policy_info) nach Aktualisierung von Police B345

policy_id	coverage	bus_start	bus_end	sys_start	sys_end	ts_id
B345	18000	2008-01-01	2009-01-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.649592000000	2010-01-31-22.31.33.495925000000

- Die Versicherungsdeckung für die Police C567 für das Jahr 2008 soll 25000 betragen:

```
UPDATE policy_info
   FOR PORTION OF BUSINESS_TIME FROM '2008-01-01' TO '2009-01-01'
   SET coverage = 25000
   WHERE policy_id = 'C567';
```

Die Aktualisierung der Police C567 gilt für den BUSINESS_TIME-Zeitraum von 2008-01-01 bis 2009-01-01. In der Tabelle policy_info gibt es nur eine Zeile für policy_id C567, die diesen Zeitraum enthält. Der BUSINESS_TIME-Zeitraum liegt innerhalb der Spaltenwerte von bus_start und bus_end für diese Zeile. Dadurch passiert Folgendes:

1. Der Deckungswert für die Zeile mit policy_id C567 wird auf 25000 aktualisiert.
2. Die Spaltenwerte von bus_start und bus_end werden nicht geändert.
3. Der Datenbankmanager aktualisiert die Werte von sys_start und ts_id auf das Datum der Aktualisierung.
4. Die ursprüngliche Zeile wird in die Protokolltabelle versetzt. Der Datenbankmanager aktualisiert den Wert für sys_end auf das Datum der Aktualisierung.

Die folgenden Tabellen geben Aufschluss über die Aktualisierung für die Police C567.

Tabelle 65. Bitemporale Tabelle (policy_info) nach Aktualisierung von Police C567

policy_id	coverage	bus_start	bus_end	sys_start	sys_end	ts_id
A123	12000	2008-01-01	2008-07-01	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
A123	16000	2008-07-01	2009-01-01	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
B345	18000	2008-03-01	2009-01-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000
C567	25000	2008-01-01	2009-01-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000

Tabelle 66. Protokolltabelle (hist_policy_info) nach Aktualisierung von Police C567

policy_id	coverage	bus_start	bus_end	sys_start	sys_end	ts_id
B345	18000	2008-01-01	2009-01-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.649592000000	2010-01-31-22.31.33.495925000000
C567	20000	2008-01-01	2009-01-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.649592000000	2010-01-31-22.31.33.495925000000

- Die Versicherungsdeckung für die Police A123 zeigt eine Erhöhung von 12000 auf 16000 am 7. Juli (2008-07-01), aber eine frühere Erhöhung auf 14000 ist nicht vorhanden:

```
UPDATE policy_info
   FOR PORTION OF BUSINESS_TIME FROM '2008-06-01' TO '2008-08-01'
   SET coverage = 14000
   WHERE policy_id = 'A123';
```

  Die Aktualisierung der Police A123 gilt für den BUSINESS_TIME-Zeitraum von 2008-06-01 bis 2008-08-01. In der Tabelle `policy_info` gibt es zwei Zeilen für `policy_id` A123, die einen Teil des Aktualisierungszeitraums umfassen.

  1. Der BUSINESS_TIME-Zeitraum ist in der Zeile mit einem `bus_start`-Wert von 2008-01-01 und einem `bus_end`-Wert von 2008-07-01 teilweise enthalten. Diese Zeile überschneidet sich mit dem Beginn des angegebenen Zeitraums, da der früheste Zeitwert im BUSINESS_TIME-Zeitraum höher als der `bus_start`-Wert, aber kleiner als der `bus_end`-Wert der Zeile ist.
  2. Der BUSINESS_TIME-Zeitraum ist in der Zeile mit einem `bus_start`-Wert von 2008-07-01 und einem `bus_end`-Wert von 2009-01-01 teilweise enthalten. Diese Zeile überschneidet sich mit dem Ende des angegebenen Zeitraums, da der späteste Zeitwert im BUSINESS_TIME-Zeitraum höher als der `bus_start`-Wert, aber kleiner als der `bus_end`-Wert der Zeile ist.

  Dadurch passiert Folgendes:

  1. Wenn sich der Wert von `bus_end` mit dem Beginn des angegebenen Zeitraums überschneidet, wird die Zeile auf den neuen Deckungswert von 14000 aktualisiert. In dieser aktualisierten Zeile ist `bus_start` auf den Wert 2008-06-01 gesetzt, der dem Anfangswert des über die Anweisung UPDATE angege-

benen Zeitraums entspricht; der Wert von bus_end wird nicht geändert. Mit den ursprünglichen Werten der Zeile wird eine zusätzliche Zeile einfügt, der Wert von bus_end wird jedoch auf 2008-06-01 gesetzt. Diese neue Zeile gibt den BUSINESS_TIME-Zeitraum an, in dem der Deckungswert 12000 betrug-.Die Einträge der Spalten sys_start, sys_end und ts_id werden vom Datenbankmanager generiert.

2. Wenn sich der Wert von bus_start mit dem Ende des angegebenen Zeitraums überschneidet, wird die Zeile auf den neuen Deckungswert von 14000 aktualisiert. In dieser aktualisierten Zeile wird der Wert von bus_start nicht geändert; bus_end wird auf den Wert 2008-08-01 gesetzt, der dem Endwert des über die Anweisung UPDATE angegebenen Zeitraums entspricht. Mit den ursprünglichen Werten der Zeile wird eine zusätzliche Zeile einfügt, der Wert von bus_start wird jedoch auf 2008-08-01 gesetzt. Diese neue Zeile gibt den BUSINESS_TIME-Zeitraum an, in dem der Deckungswert 16000 betrug-.Die Einträge der Spalten sys_start, sys_end und ts_id werden vom Datenbankmanager generiert.
3. Die ursprünglichen Zeilen werden in die Protokolltabelle versetzt. Der Datenbankmanager aktualisiert den Wert für sys_end auf das Datum der Aktualisierung.

Die folgenden Tabellen geben Aufschluss über die Aktualisierung für die Police A123.

Tabelle 67. Bitemporale Tabelle (policy_info) nach Aktualisierung von Police A123

policy_id	coverage	bus_start	bus_end	sys_start	sys_end	ts_id
A123	12000	2008-01-01	2008-06-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000
A123	14000	2008-06-01	2008-07-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000
A123	14000	2008-07-01	2008-08-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000
A123	16000	2008-08-01	2009-01-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000
B345	18000	2008-03-01	2009-01-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000
C567	25000	2008-01-01	2009-01-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000

Tabelle 68. Protokolltabelle (hist_policy_info) nach Aktualisierung von Police A123

policy_id	coverage	bus_start	bus_end	sys_start	sys_end	ts_id
A123	12000	2008-01-01	2008-07-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.649592000000	2010-01-31-22.31.33.495925000000
A123	16000	2008-07-01	2009-01-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.649592000000	2010-01-31-22.31.33.495925000000

Tabelle 68. Protokolltabelle (hist_policy_info) nach Aktualisierung von Police A123 (Forts.)

policy_id	coverage	bus_start	bus_end	sys_start	sys_end	ts_id
B345	18000	2008-01-01	2009-01-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.649592000000	2010-01-31-22.31.33.495925000000
C567	20000	2008-01-01	2009-01-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.649592000000	2010-01-31-22.31.33.495925000000

Löschen von Daten in bitemporaler Tabelle

Das Löschen von Daten aus einer bitemporalen Tabelle führt dazu, dass Zeilen aus der Tabelle gelöscht, Zeilen zu der zugehörigen Protokolltabelle hinzugefügt und potenziell neue Zeilen in die bitemporale Tabelle eingefügt werden.

Informationen zu diesem Vorgang

Neben der regulären DELETE-Anweisung unterstützen bitemporale Tabellen auch zeitraumbezogene Aktualisierungen, bei denen die DELETE-Anweisung die Klausel FOR PORTION OF BUSINESS_TIME enthält. Eine Zeile ist ein Kandidat für einen Löschvorgang, wenn die zugehörige Beginn- und/oder Endspalte für einen Zeitraum in den Zeitraum fällt, der in der Klausel FOR PORTION OF BUSINESS_TIME angegeben ist. Alle vorhandenen betroffenen Zeilen werden in die Protokolltabelle kopiert, bevor sie gelöscht werden.

Vorgehensweise

Zum Löschen von Daten aus einer bitemporalen Tabelle wird die Anweisung DELETE FROM verwendet. Es wurde beispielsweise festgestellt, dass die Police A123 zwischen dem 15. Juni 2008 und dem 15. August 2008 über keine Deckung verfügt hat. Die Daten wurden am 1. September 2011 (2011-09-01) aus der Tabelle gelöscht, die im Abschnitt „ Aktualisieren von Daten in einer bitemporalen Tabelle" aktualisiert wurde.

```
DELETE FROM policy_info
   FOR PORTION OF BUSINESS_TIME FROM '2008-06-15' TO '2008-08-15'
   WHERE policy_id = 'A123';
```

Ergebnisse

Die ursprünglichen Daten aus den Tabellen `policy_info` und `hist_policy_info` lauten wie folgt:

Tabelle 69. Daten in der bitemporalen Tabelle (policy_info) vor der DELETE-Anweisung

policy_id	coverage	bus_start	bus_end	sys_start	sys_end	ts_id
A123	12000	2008-01-01	2008-06-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000
A123	14000	2008-06-01	2008-07-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000
A123	14000	2008-07-01	2008-08-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000

Tabelle 69. Daten in der bitemporalen Tabelle (policy_info) vor der DELETE-Anweisung (Forts.)

policy_id	coverage	bus_start	bus_end	sys_start	sys_end	ts_id
A123	16000	2008-08-01	2009-01-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000
B345	18000	2008-03-01	2009-01-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000
C567	25000	2008-01-01	2009-01-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000

Tabelle 70. Daten in der Protokolltabelle (hist_policy_info) vor der DELETE-Anweisung

policy_id	coverage	bus_start	bus_end	sys_start	sys_end	ts_id
A123	12000	2008-01-01	2008-07-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.649592000000	2010-01-31-22.31.33.495925000000
A123	16000	2008-07-01	2009-01-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.649592000000	2010-01-31-22.31.33.495925000000
B345	18000	2008-01-01	2009-01-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.649592000000	2010-01-31-22.31.33.495925000000
C567	20000	2008-01-01	2009-01-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.649592000000	2010-01-31-22.31.33.495925000000

Das Löschen von Daten aus einer bitemporalen Tabelle unter Verwendung der Klausel FOR PORTION OF BUSINESS_TIME führt dazu, dass Zeilen gelöscht werden und möglicherweise Zeilen eingefügt werden, wenn der Zeitraum für eine Zeile einen Teil des in der Anweisung DELETE FROM angegebenen Zeitraums abdeckt. Das Löschen von Daten im Zusammenhang mit der Police A123 bezieht sich auf den BUSINESS_TIME-Zeitraum vom 15.06.2008 (2008-06-15) bis zum 15.08.2008 (2008-08-15). In der Tabelle `policy_info` befinden sich drei Zeilen für `policy_id` A123, die diesen Zeitraum vollständig oder teilweise abdecken.

Dadurch passiert Folgendes:

- Es gibt eine Zeile, für die der BUSINESS_TIME-Zeitraum in der Anweisung DELETE FROM den gesamten Zeitraum für eine Zeile abdeckt. Die Zeile mit einem `bus_start`-Wert von 2008-07-01 und einem `bus_end`-Wert von 2008-08-01 wird gelöscht.
- Wenn nur der `bus_end`-Wert innerhalb des angegebenen Zeitraums liegt, wird die Zeile gelöscht. Eine neue Zeile mit den ursprünglichen Werten aus der gelöschten Zeile wird eingefügt, wobei der `bus_end`-Wert allerdings auf 2008-06-15 gesetzt ist. Die Einträge der Spalten `sys_start`, `sys_end` und `ts_id` werden vom Datenbankmanager generiert.
- Wenn nur der `bus_start`-Wert innerhalb des angegebenen Zeitraums liegt, wird die Zeile gelöscht. Eine neue Zeile mit den ursprünglichen Werten aus der gelöschten Zeile wird eingefügt, wobei der `bus_start`-Wert allerdings auf 2008-08-15 gesetzt ist. Die Einträge der Spalten `sys_start`, `sys_end` und `ts_id` werden vom Datenbankmanager generiert.

- Die ursprünglichen Zeilen werden in die Protokolltabelle versetzt. Der Datenbankmanager aktualisiert den Wert für sys_end auf das Löschdatum.

Tabelle 71. Daten in der bitemporalen Tabelle (policy_info) nach der DELETE-Anweisung

policy_id	coverage	bus_start	bus_end	sys_start	sys_end	ts_id
A123	12000	2008-01-01	2008-06-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000
A123	14000	2008-06-01	2008-06-15	2011-09-01-12.18.22.959254000000	9999-12-30-00.00.00.000000000000	2011-09-01-12.18.22.959254000000
A123	16000	2008-08-15	2009-01-01	2011-09-01-12.18.22.959254000000	9999-12-30-00.00.00.000000000000	2011-09-01-12.18.22.959254000000
B345	18000	2008-03-01	2009-01-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000
C567	25000	2008-01-01	2009-01-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000

Tabelle 72. Protokolltabelle (hist_policy_info) nach der DELETE-Anweisung

policy_id	coverage	bus_start	bus_end	sys_start	sys_end	ts_id
A123	12000	2008-01-01	2008-07-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.649592000000	2010-01-31-22.31.33.495925000000
A123	16000	2008-07-01	2009-01-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.649592000000	2010-01-31-22.31.33.495925000000
B345	18000	2008-01-01	2009-01-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.649592000000	2010-01-31-22.31.33.495925000000
C567	20000	2008-01-01	2009-01-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.649592000000	2010-01-31-22.31.33.495925000000
A123	14000	2008-06-01	2008-07-01	2011-02-28-09.10.12.649592000000	2011-09-01-12.18.22.959254000000	2011-09-01-12.18.22.959254000000
A123	14000	2008-07-01	2008-08-01	2011-02-28-09.10.12.649592000000	2011-09-01-12.18.22.959254000000	2011-09-01-12.18.22.959254000000
A123	16000	2008-08-01	2009-01-01	2011-02-28-09.10.12.649592000000	2011-09-01-12.18.22.959254000000	2011-09-01-12.18.22.959254000000

Abfragen von bitemporalen Daten

Durch das Abfragen einer bitemporalen Tabelle können Ergebnisse für einen bestimmten Zeitraum zurückgegeben werden. Diese Ergebnisse können aktuelle Werte, früher protokollierte Werte und zukünftige Werte enthalten.

Informationen zu diesem Vorgang

Beim Abfragen einer bitemporalen Tabelle kann FOR BUSINESS_TIME und/oder FOR SYSTEM_TIME in die FROM-Klausel eingefügt werden. Mithilfe dieser Zeitraumangaben kann der aktuelle, der frühere und der zukünftige Status der Daten abgefragt werden. Die Zeiträume werden wie folgt angegeben:

AS OF *wert1*
: Enthält alle Zeilen, bei denen der Anfangswert für den Zeitraum kleiner-gleich *wert1* und der Endwert für den Zeitraum größer als *wert1* ist. Dies ermöglicht Ihnen, Ihre Daten ab einem bestimmten Zeitpunkt abzufragen.

FROM *wert1* **TO** *wert2*
: Enthält alle Zeilen, bei denen der Anfangswert für den Zeitraum größer-gleich *wert1* und der Endwert für den Zeitraum kleiner als *wert2* ist. Dies bedeutet, dass die Anfangszeit in den Zeitraum eingeschlossen ist, die Endzeit jedoch nicht.

BETWEEN *wert1* **AND** *wert2*
: Enthält alle Zeilen, bei denen sich ein beliebiger Zeitraum mit einem beliebigen Zeitpunkt zwischen *wert1* und *wert2* überschneidet. Eine Zeile wird zurückgegeben, wenn der Anfangswert für den Zeitraum kleiner-gleich *wert2* und der Endwert für den Zeitraum größer als *wert1* ist.

Im folgenden Abschnitt sind einige Beispielabfragen aufgelistet.

Vorgehensweise

Zum Abfragen einer bitemporalen Tabelle wird die Anweisung SELECT verwendet. Beispielsweise fordert jede der folgenden Abfragen Policeninformationen zu `policy_id` A123 aus den im Abschnitt „ Löschen von Daten aus einer bitemporalen Tabelle" aufgeführten Ergebnistabellen an. Jede Abfrage verwendet eine Variante der Zeitraumangabe.
Die Tabelle `policy_info` und die zugehörige Protokolltabelle sehen wie folgt aus:

Tabelle 73. Bitemporale Tabelle: policy_info

policy_id	coverage	bus_start	bus_end	sys_start	sys_end	ts_id
A123	12000	2008-01-01	2008-06-01	2011-02-28-09.10.12.64959200000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.64959200000
A123	14000	2008-06-01	2008-06-15	2011-09-01-12.18.22.959254000000	9999-12-30-00.00.00.000000000000	2011-09-01-12.18.22.959254000000
A123	16000	2008-08-15	2009-01-01	2011-09-01-12.18.22.959254000000	9999-12-30-00.00.00.000000000000	2011-09-01-12.18.22.959254000000
B345	18000	2008-03-01	2009-01-01	2011-02-28-09.10.12.64959200000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.64959200000
C567	25000	2008-01-01	2009-01-01	2011-02-28-09.10.12.64959200000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.64959200000

Tabelle 74. Protokolltabelle: hist_policy_info

policy_id	coverage	bus_start	bus_end	sys_start	sys_end	ts_id
A123	12000	2008-01-01	2008-07-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.64959200000	2010-01-31-22.31.33.495925000000
A123	16000	2008-07-01	2009-01-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.64959200000	2010-01-31-22.31.33.495925000000
B345	18000	2008-01-01	2009-01-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.64959200000	2010-01-31-22.31.33.495925000000
C567	20000	2008-01-01	2009-01-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.64959200000	2010-01-31-22.31.33.495925000000
A123	14000	2008-06-01	2008-07-01	2011-02-28-09.10.12.64959200000	2011-09-01-12.18.22.959254000000	2011-09-01-12.18.22.959254000000
A123	14000	2008-07-01	2008-08-01	2011-02-28-09.10.12.64959200000	2011-09-01-12.18.22.959254000000	2011-09-01-12.18.22.959254000000
A123	16000	2008-08-01	2009-01-01	2011-02-28-09.10.12.64959200000	2011-09-01-12.18.22.959254000000	2011-09-01-12.18.22.959254000000

- Abfrage ohne Zeitraumangabe. Beispiel:

```
SELECT policy_id, coverage, bus_start, bus_end
   FROM policy_info
   where policy_id = 'A123'
```

  Diese Abfrage gibt drei Zeilen zurück. Von der Anweisung SELECT wird nur die Tabelle `policy_info` abgefragt. Die Protokolltabelle wird nicht abgefragt, da FOR SYSTEM_TIME nicht angegeben wurde.

```
A123, 12000, 2008-01-01, 2008-06-01
A123, 14000, 2008-06-01, 2008-06-15
A123, 16000, 2008-08-15, 2009-01-01
```

- Abfrage mit Angabe von FOR SYSTEM_TIME FROM...TO. Beispiel:

```
SELECT policy_id, coverage, bus_start, bus_end
   FROM policy_info
   FOR SYSTEM_TIME FROM
      '0001-01-01-00.00.00.000000' TO '9999-12-30-00.00.00.000000000000'
   where policy_id = 'A123'
```

  Diese Abfrage gibt acht Zeilen zurück. Von der Anweisung SELECT wird die Tabelle `policy_info` und die Tabelle `hist_policy_info` abgefragt.

```
A123, 12000, 2008-01-01, 2008-06-01
A123, 14000, 2008-06-01, 2008-06-15
A123, 16000, 2008-08-15, 2009-01-01
A123, 12000, 2008-01-01, 2008-07-01
A123, 16000, 2008-07-01, 2009-01-01
A123, 14000, 2008-06-01, 2008-07-01
A123, 14000, 2008-07-01, 2008-08-01
A123, 16000, 2008-08-01, 2009-01-01
```

- Abfrage mit Angabe von FOR BUSINESS_TIME AS OF. Beispiel:

```
SELECT policy_id, coverage, bus_start, bus_end
   FROM policy_info
   FOR BUSINESS_TIME AS OF '2008-07-15'
   where policy_id = 'A123'
```

Diese Abfrage gibt keine Zeilen zurück. Von der Anweisung SELECT wird nur die Tabelle `policy_info` abgefragt und es gibt keine Zeilen für A123, bei denen der Anfangswert für den Zeitraum kleiner-gleich *2008-07-15* und der Endwert für den Zeitraum größer als *2008-07-15* ist. Die Police A123 verfügte am 15.07.2008 (2008-07-15) über keine Deckung. Die Protokolltabelle wird nicht abgefragt, da FOR SYSTEM_TIME nicht angegeben wurde.

- Abfrage mit Angabe von FOR BUSINESS_TIME AS OF und FOR SYSTEM_TIME FROM...TO. Beispiel:

```
SELECT policy_id, coverage, bus_start, bus_end
   FROM policy_info
   FOR BUSINESS_TIME AS OF '2008-07-15'
   FOR SYSTEM_TIME FROM
      '0001-01-01-00.00.00.000000' TO '9999-12-30-00.00.00.000000000000'
   where policy_id = 'A123'
```

Diese Abfrage gibt zwei Zeilen zurück. Von der Anweisung SELECT wird die Tabelle `policy_info` und die Tabelle `hist_policy_info` abgefragt. Die zurückgegebenen Zeilen befinden sich in die Protokolltabelle.

```
A123, 16000, 2008-07-01, 2009-01-01
A123, 14000, 2008-07-01, 2008-08-01
```

Szenarios und Beispiele für Tabellen

Dieser Abschnitt enthält Beschreibungen von Szenarios und Beispielen für Tabellen.

Szenarios: Optimistisches Sperren und zeitbasierte Erkennung

Drei Szenarios werden bereitgestellt, die zeigen, wie Sie das optimistische Sperren in Ihren Anwendungen mit und ohne zeitbasierte Erkennung sowie mit und ohne implizit verdeckte Spalten aktivieren und implementieren.

Szenario: Verwenden des optimistischen Sperrens in einem Anwendungsprogramm

Dieses Szenario veranschaulicht, wie das optimistische Sperren in einem Anwendungsprogramm implementiert wird. Es behandelt sechs verschiedene Szenarios.

Betrachten Sie die folgende Abfolge von Ereignissen in einer Anwendung, die zur Nutzung des optimistischen Sperrens entworfen und eingerichtet wurde:

```
SELECT QUANTITY, row change token FOR STOCK, RID_BIT(STOCK)
 INTO :h_quantity, :h_rct, :h_rid
 FROM STOCK WHERE PARTNUM = 3500
```

In diesem Szenario liest die Anwendungslogik jede Zeile. Da diese Anwendung wie in „Aktivieren des optimistischen Sperrens in Anwendungen" auf Seite 367 beschrieben für das optimistische Sperren aktiviert ist, enthält die SELECT-Liste den Wert der Funktion RID_BIT(), der in der Hostvariablen :h_rid gespeichert wird, und den Wert des Zeilenänderungstokens ('row change token'), der in der Hostvariablen :h_rct gespeichert wird.

Bei aktiviertem optimistischen Sperren nimmt die Anwendung optimistischerweise an, dass alle Zeilen, die Zielzeilen für UPDATE- oder DELETE-Operationen sind,

unverändert bleiben, auch wenn sie nicht durch Sperren geschützt werden. Zur Verbesserung des gemeinsamen Zugriffs auf die Datenbank entfernt die Anwendung die Zeilensperre(n) durch eine der folgenden Methoden:

- Sie schreibt die UOW (Unit of Work, Arbeitseinheit) mit COMMIT fest, wobei die Zeilensperren entfernt werden.
- Sie schließt den Cursor mit der Klausel WITH RELEASE, wobei die Zeilensperren entfernt werden.
- Sie verwendet eine niedrigere Isolationsstufe:
  - CURSOR STABILITY (CS, Cursorstabilität), wobei die Zeile nicht gesperrt ist, nachdem der Cursor die nächste Zeile abgerufen hat bzw. bis das Ende der Ergebnistabelle erreicht ist.
  - UNCOMMITED READ (UR, nicht festgeschriebener Lesevorgang), wobei alle nicht festgeschriebenen Daten einen neuen (nicht festgeschriebenen) Wert für das Zeilenänderungstoken haben. Wenn die nicht festgeschriebenen Daten rückgängig gemacht werden (Rollback), erhält das alte, festgeschriebene Zeilenänderungstoken einen anderen Wert.

    Anmerkung: Unter der Annahme, dass Aktualisierungen im Normalfall nicht rückgängig gemacht werden, bietet die Isolationsstufe UR den meisten gemeinsamen Zugriff.
- Sie trennt die Verbindung zur Datenbank, sodass alle DB2-Serverressourcen für die Anwendung freigegeben werden. (.NET-Anwendungen verwenden diesen Modus häufig.)

Die Anwendung verarbeitet die Zeilen und entscheidet, eine von ihnen optimistisch zu aktualisieren:

```
UPDATE STOCK SET QUANTITY = QUANTITY – 1
 WHERE row change token FOR STOCK = :h_rct AND
 RID_BIT(STOCK) = :h_rid
```

Die Anweisung UPDATE aktualisiert die in der oben gezeigten SELECT-Anweisung identifizierte Zeile.

Das UPDATE-Vergleichselement mit Suche wird als Direktabruf aus der Tabelle eingeplant:

```
RID_BIT(STOCK) = :h_rid
```

Der Direktabruf ist ein sehr effizienter Zugriffsplan, dessen Aufwand vom DB2-Optimierungsprogramm problemlos kalkuliert werden kann. Wenn das Vergleichselement mit der Funktion RID_BIT() keine Zeile findet, wurde die Zeile gelöscht und die UPDATE-Operation schlägt wegen der nicht gefundenen Zeile fehl.

Vorausgesetzt, das Vergleichselement mit RID_BIT() findet eine Zeile, findet das Vergleichselement 'row change token FOR STOCK = :h_rct' die Zeile, sofern die Zeile nicht geändert wurde. Wenn sich das Zeilenänderungstoken seit Ausführung der Anweisung SELECT geändert hat, schlägt die UPDATE-Operation mit Suche aufgrund nicht gefundener Zeile fehl.

In Tabelle 75 auf Seite 443 sind die möglichen Szenarios aufgeführt, die bei aktiviertem optimistischem Sperren auftreten können.

Tabelle 75. Mögliche Szenarios bei aktiviertem optimistischem Sperren

Szenario-ID	Aktion	Ergebnis
Szenario 1	Es ist keine Spalte für die Zeilenänderungszeitmarke (ROW CHANGE TIMESTAMP) in der Tabelle definiert, und keine andere Anwendung hat die Zeile geändert.	Die UPDATE-Operation wird erfolgreich ausgeführt, da das Vergleichselement mit der Zeilenänderungszeitmarke für die durch ':h_rid' angegebene Zeile erfolgreich ausgewertet wird.
Szenario 2	In der Tabelle ist eine Zeilenänderungszeitmarke (ROW CHANGE TIMESTAMP) definiert. Eine andere Anwendung aktualisiert die Zeile nach der SELECT-Operation und vor der UPDATE-Operation (und der COMMIT-Operation), wobei auch die Spalte für die Zeilenänderungszeitmarke aktualisiert wird.	Der Vergleich des Vergleichselements mit dem Zeilenänderungstoken zwischen dem Wert des Tokens, der aus der Zeitmarke in der Zeile zum Zeitpunkt der SELECT-Operation generiert wurde, und dem Zeitmarkenwert des Tokens, der aktuell in der Zeile enthalten ist, schlägt fehl. Infolgedessen kann die Anweisung UPDATE keine Zeile finden.
Szenario 3	In der Tabelle ist eine Zeilenänderungszeitmarke (ROW CHANGE TIMESTAMP) definiert. Eine andere Anwendung aktualisiert die Zeile, sodass die Zeile ein neues Zeilenänderungstoken erhält. Diese Anwendung wählt die Zeile mit der Isolationsstufe UR (Uncommitted Read, nicht festgeschriebener Lesevorgang) aus und ruft das neue, nicht festgeschriebene Zeilenänderungstoken ab.	Diese Anwendung führt die UPDATE-Operation aus, die wartet, bis die andere Anwendung die Zeilensperre freigibt. Das Vergleichselement mit dem Zeilenänderungstoken wird erfolgreich ausgewertet, wenn die andere Anwendung die Änderung mit dem neuen Token festschreibt (COMMIT), sodass die UPDATE-Operation erfolgreich ausgeführt wird. Der Vergleich durch das Vergleichselement mit dem Zeilenänderungstoken schlägt fehl, wenn die andere Anwendung die Änderung der Zeile auf den vorherigen Stand mit dem alten Token rückgängig macht (Rollback), sodass die UPDATE-Operation keine Zeile finden kann.
Szenario 4	Es ist keine Spalte für die Zeilenänderungszeitmarke (ROW CHANGE TIMESTAMP) in der Tabelle definiert. Eine andere Zeile wird auf derselben Seite nach der SELECT-Operation, jedoch vor der UPDATE-Operation aktualisiert, gelöscht oder eingefügt.	Der Vergleich durch das Vergleichselement mit dem Zeilenänderungstoken schlägt fehl, weil sich der Wert des Zeilenänderungstokens für alle Zeile auf der Seite geändert hat, sodass die UPDATE-Anweisung keine Zeile finden kann, selbst wenn die fragliche Zeile selbst in Wirklichkeit nicht geändert wurde. Dieses Szenario eines falschen Negativwerts würde nicht zu einem UPDATE-Fehler führen, wenn eine Spalte für die Zeilenänderungszeitmarke hinzugefügt worden wäre.
Szenario 5	Die Tabelle wurde geändert, sodass sie eine Spalte für die Zeilenänderungszeitmarke enthält, und die Zeile, die von der Anweisung SELECT zurückgegeben wurde, wurde seit dem Zeitpunkt der Anweisung ALTER TABLE nicht geändert. Eine andere Anwendung aktualisiert die Zeile, indem sie die Spalte für die Zeilenänderungszeitmarke dieser Zeile im Prozess mit der aktuellen Zeitmarke hinzufügt.	Der Vergleich durch das Vergleichselement mit dem Zeilenänderungstoken zwischen dem zuvor generierten Token und dem Tokenwert aus der Spalte der Zeilenänderungszeitmarke schlägt fehl, sodass die Anweisung UPDATE keine Zeile finden kann. Da die fragliche Zeile tatsächlich geändert wurde, handelt es sich in diesem Fall nicht um ein Szenario mit falschem Negativwert.

Tabelle 75. Mögliche Szenarios bei aktiviertem optimistischem Sperren (Forts.)

Szenario-ID	Aktion	Ergebnis
Szenario 6	Die Tabelle wird nach der SELECT-Operation und vor der UPDATE-Operation reorganisiert. Die Satz-ID (RID), die durch ':h_rid' angegeben wird, findet keine Zeile bzw. enthält eine Zeile mit einem anderen Token, sodass die Aktualisierung fehlschlägt. Dies ist die Form eines falschen Negativwerts, die sich nicht vermeiden lässt, selbst wenn eine Spalte für die Zeilenänderungszeitmarke in der Zeile vorhanden ist.	Die Zeile selbst wird durch die Reorganisation nicht aktualisiert. Der RID_BIT-Teil des Vergleichselements kann jedoch die ursprüngliche Zeile nach der Reorganisation nicht mehr identifizieren.

Szenarios: Optimistisches Sperren mit implizit verdeckten Spalten

Die folgenden Szenarios veranschaulichen, wie das optimistische Sperren in einem Anwendungsprogramm unter Verwendung implizit verdeckter Spalten, das heißt, mit Spalten, die mit dem Attribut IMPLICITLY HIDDEN definiert sind, implementiert wird.

Nehmen Sie für diese Szenarios an, dass die Tabelle SALARY_INFO mit drei Spalten definiert ist. Die erste Spalte ist eine implizit verdeckte Spalte für die Zeilenänderungszeitmarke (ROW CHANGE TIMESTAMP), deren Werte immer generiert werden.

Szenario 1:

In der folgenden Anweisung wird die implizit verdeckte Spalte in der Spaltenliste explizit aufgeführt und ein Wert für sie in der Klausel VALUES vorgesehen:

```
INSERT INTO SALARY_INFO (UPDATE_TIME, LEVEL, SALARY)
  VALUES (DEFAULT, 2, 30000)
```

Szenario 2:

Die folgende Anweisung INSERT enthält eine implizite Spaltenliste. Eine implizite Spaltenliste umfasst keine implizit verdeckten Spalten. Daher enthält die Klausel VALUES nur Werte für die beiden anderen Spalten:

```
INSERT INTO SALARY_INFO
  VALUES (2, 30000)
```

In diesem Fall muss die Spalte UPDATE_TIME so definiert sein, dass sie einen Standardwert besitzt. Dieser Standardwert für die Zeile verwendet, die eingefügt wird.

Szenario 3:

In der folgenden Anweisung wird die implizit verdeckte Spalte explizit in der SELECT-Liste aufgeführt, und in der Ergebnismenge wird ein Wert für sie angezeigt:

```
SELECT UPDATE_TIME, LEVEL, SALARY FROM SALARY_INFO
  WHERE LEVEL = 2

  UPDATE_TIME                   LEVEL    SALARY
  -------------------------- ----------- -----------
  2006-11-28-10.43.27.560841      2      30000
```

Szenario 4:

In der folgenden Anweisung wird die Spaltenliste implizit durch die Notation '*' generiert, sodass die implizit verdeckte Spalte in der Ergebnismenge nicht auftritt:

```
SELECT * FROM SALARY_INFO
  WHERE LEVEL = 2

  LEVEL       SALARY
  ----------- -----------
       2      30000
```

Szenario 5:

In der folgenden Anweisung wird die Spaltenliste implizit durch die Verwendung der Notation '*' generiert. Der Wert der implizit verdeckten Spalte wird durch die Verwendung des Ausdrucks ROW CHANGE TIMESTAMP FOR ebenfalls zurückgegeben:

```
SELECT ROW CHANGE TIMESTAMP FOR SALARY_INFO AS ROW_CHANGE_STAMP, SALARY_INFO.*
  FROM SALARY_INFO WHERE LEVEL = 2
```

Die Ergebnistabelle ist der in Szenario 3 ähnlich (die Spalte UPDATE_TIME wird als ROW_CHANGE_STAMP zurückgegeben).

Szenario: Zeitbasierte Aktualisierungserkennung

Dieses Szenario veranschaulicht, wie das optimistische Sperren in einem Anwendungsprogramm unter Verwendung der Aktualisierungserkennung mithilfe einer Zeitmarke implementiert wird. Es behandelt drei verschiedene Szenarios.

In diesem Szenario wählt die Anwendung alle Zeilen aus, die in den letzten 30 Tagen geändert wurden.

```
SELECT * FROM TAB WHERE
  ROW CHANGE TIMESTAMP FOR TAB <=
  CURRENT TIMESTAMP AND
  ROW CHANGE TIMESTAMP FOR TAB >=
  CURRENT TIMESTAMP - 30 days;
```

Szenario 1:

Es ist keine Spalte für die Zeilenänderungszeitmarke (ROW CHANGE TIMESTAMP) in der Tabelle definiert. Die Ausführung der Anweisung schlägt mit SQL20431N fehl. Dieser SQL-Ausdruck wird nur für Tabellen mit einer definierten Spalte für die Zeilenänderungszeitmarke unterstützt.

Anmerkung: Dieses Szenario funktioniert unter z/OS.

Szenario 2:

Eine Spalte für die Zeilenänderungszeitmarke (ROW CHANGE TIMESTAMP) wurde beim Erstellen der Tabelle definiert:

```
CREATE TABLE TAB ( ..., RCT TIMESTAMP NOT NULL
                        GENERATED ALWAYS
                        FOR EACH ROW ON UPDATE AS
                        ROW CHANGE TIMESTAMP)
```

Diese Anweisung gibt alle Zeilen zurück, die in den letzten 30 Tagen eingefügt oder aktualisiert wurden.

Szenario 3:

Eine Spalte für die Zeilenänderungszeitmarke wurde der Tabelle zu einem Zeitpunkt während der letzten 30 Tage mithilfe der Anweisung ALTER TABLE hinzugefügt:

```
ALTER TABLE TAB ADD COLUMN RCT TIMESTAMP NOT NULL
                               GENERATED ALWAYS
                               FOR EACH ROW ON UPDATE AS
                               ROW CHANGE TIMESTAMP
```

Diese Anweisung gibt alle Zeilen in der Tabelle zurück. Alle Zeilen, die seit Ausführung der Anweisung ALTER TABLE nicht geändert wurden, verwenden als Standardwert die Zeitmarke der Anweisung ALTER TABLE selbst, während alle anderen Zeilen, die seit dem geändert wurden, eine eindeutige Zeitmarke haben.

Kapitel 13. Integritätsbedingungen

In jeder Geschäftsumgebung müssen Daten häufig bestimmten Rahmenbedingungen oder Regeln genügen. Zum Beispiel muss eine Personalnummer eindeutig sein. Der Datenbankmanager stellt *Integritätsbedingungen* bereit, die zur Umsetzung solcher Regeln verwendet werden können.

Die folgenden Typen von Integritätsbedingungen sind verfügbar:

- NOT NULL, Integritätsbedingung
- Eindeutige Integritätsbedingungen (oder eindeutige Integritätsbedingungen über Schlüssel)
- Integritätsbedingungen über Primärschlüssel
- Integritätsbedingungen über Fremdschlüssel (oder referenzielle Integritätsbedingungen)
- Prüfungen auf Integritätsbedingungen (in Tabellen)
- Informative Integritätsbedingungen

Integritätsbedingungen werden nur Tabellen zugeordnet; sie werden entweder im Rahmen des Tabellenerstellungsprozesses (mithilfe der Anweisung CREATE TABLE) erstellt, oder sie werden einer Tabellendefinition nach der Erstellung der Tabelle (mithilfe der Anweisung ALTER TABLE) hinzugefügt. Mithilfe der Anweisung ALTER TABLE können Sie Integritätsbedingungen ändern. In den meisten Fällen, können vorhandene Integritätsbedingungen jederzeit gelöscht werden; diese Aktion beeinflusst die Struktur der Tabelle oder die darin gespeicherten Daten nicht.

Anmerkung: Eindeutige Integritätsbedingungen über Schlüssel und Integritätsbedingungen über Primärschlüssel werden nur Tabellenobjekten zugeordnet; sie werden häufig durch die Verwendung eines oder mehrerer Indizes für den eindeutigen Schlüssel oder den Primärschlüssel erzwungen.

Typen von Integritätsbedingungen

Eine *Integritätsbedingung* ist eine Regel, die zu Optimierungszwecken verwendet wird.

Es gibt fünf Typen von Integritätsbedingungen:

- Bei der *Integritätsbedingung NOT NULL* handelt es sich um eine Regel, die vermeidet, dass in eine oder mehrere Spalten innerhalb einer Tabelle Nullwerte eingegeben werden.
- Eine *eindeutige Integritätsbedingung* (auch als *Integritätsbedingung über eindeutige Schlüssel* bezeichnet) ist eine Regel, die untersagt, dass in einer oder mehreren Spalten innerhalb einer Tabelle doppelte Werte auftreten. Eindeutige Integritätsbedingungen werden in Form von eindeutigen Schlüsseln und Primärschlüsseln unterstützt. Zum Beispiel kann eine eindeutige Integritätsbedingung für die Lieferantenkennung einer Lieferantentabelle definiert werden, um sicherzustellen, dass nicht ein und dieselbe Kennung zwei Lieferanten zugewiesen wird.
- Eine *Integritätsbedingung über Primärschlüssel* ist eine Spalte oder eine Kombination von Spalten, die die gleichen Eigenschaften wie eine eindeutige Integritätsbedingung besitzt. Ein Primärschlüssel und Integritätsbedingungen über Fremdschlüssel dienen zur Definition von Beziehungen zwischen Tabellen.

- Eine *Integritätsbedingung über Fremdschlüssel* (auch als *referenzielle Integritätsbedingung* bezeichnet) ist eine logische Regel über Werte in einer oder mehreren Spalten in mindestens einer Tabelle. Zum Beispiel enthält eine Gruppe von Tabellen gemeinsame Informationen über die Lieferanten einer Firma. Gelegentlich ändert sich der Name eines Lieferanten. Sie können eine referenzielle Integritätsbedingung definieren, die besagt, dass die Kennung (ID) des Lieferanten in einer Tabelle mit einer Lieferanten-ID in den Lieferanteninformationen übereinstimmen muss. Diese Integritätsbedingung verhindert Einfüge-, Aktualisierungs- oder Löschoperationen, die ansonsten zu fehlenden Lieferanteninformationen führen würden.
- Eine *Prüfung auf Integritätsbedingung (in Tabellen)* (oder auch *Prüfung auf Integritätsbedingung*) definiert Einschränkungen für Daten, die einer bestimmten Tabelle hinzugefügt werden. Zum Beispiel kann eine Prüfung auf Integritätsbedingung in einer Tabelle gewährleisten, dass das Gehaltsniveau eines Mitarbeiters stets mindestens $20.000 beträgt, wenn Gehaltsdaten in einer Tabelle mit Personalinformationen hinzugefügt oder aktualisiert werden.

Eine *informative Integritätsbedingung* ist ein Attribut eines bestimmten Typs von Integritätsbedingung, das nicht vom Datenbankmanager umgesetzt wird.

NOT NULL, Integritätsbedingung

Integritätsbedingungen NOT NULL verhindern, dass Nullwerte in eine Spalte eingegeben werden.

Der Nullwert dient in Datenbanken zur Darstellung eines unbekannten Status. Standardmäßig unterstützen alle integrierten Datentypen, die im Datenbankmanager verfügbar sind, das Vorhandensein von Nullwerten. Einige Geschäftsregeln verlangen jedoch möglicherweise, dass in jedem Fall ein Wert angegeben werden muss (z. B. muss jeder Mitarbeiter Kontaktinformationen für den Notfall angeben). Die Integritätsbedingung NOT NULL dient zur Sicherstellung, dass einer bestimmten Spalte einer Tabelle nie der Nullwert zugeordnet wird. Wenn die Integritätsbedingung NOT NULL für eine bestimmte Spalte definiert wurde, schlägt jede Einfüge- oder Aktualisierungsoperation fehl, die versucht, in dieser Spalte einen Nullwert zu speichern.

Da Integritätsbedingungen nur für eine bestimmte Tabelle gelten, werden sie in der Regel zusammen mit den Attributen einer Tabelle während des Tabellenerstellungsprozesses definiert. Die folgende Anweisung CREATE TABLE zeigt, wie die Integritätsbedingung NOT NULL für eine bestimmte Spalte definiert wird:

```
CREATE TABLE EMPLOYEES    (. . .
                           EMERGENCY_PHONE    CHAR(14)  NOT NULL,
                           . . .
                           );
```

Eindeutige Integritätsbedingungen

Eindeutige Integritätsbedingungen stellen sicher, dass die Werte in einer Gruppe von Spalten für alle Zeilen in der Tabelle eindeutig und nicht null sind. Die Spalten, die in einer eindeutigen Integritätsbedingung angegeben werden, müssen mit NOT NULL (keine Nullwerte möglich) definiert werden. Der Datenbankmanager stellt die Eindeutigkeit des Schlüssels bei Änderungen an den Spalten der eindeutigen Integritätsbedingung mithilfe eines eindeutigen Index sicher.

Eindeutige Integritätsbedingungen können in der Anweisung CREATE TABLE oder ALTER TABLE durch die Klausel UNIQUE definiert werden. Zum Beispiel könnte

eine typische eindeutige Integritätsbedingung in einer Tabelle DEPARTMENT mit Daten über Abteilungen darin bestehen, dass die Abteilungsnummer (DEPTNO) eindeutig und nicht null ist.

In Abb. 38 sehen Sie, dass ein doppelter Datensatz nicht zu einer Tabelle hinzugefügt wird, wenn für die Tabelle eine eindeutige Integritätsbedingung vorhanden ist:

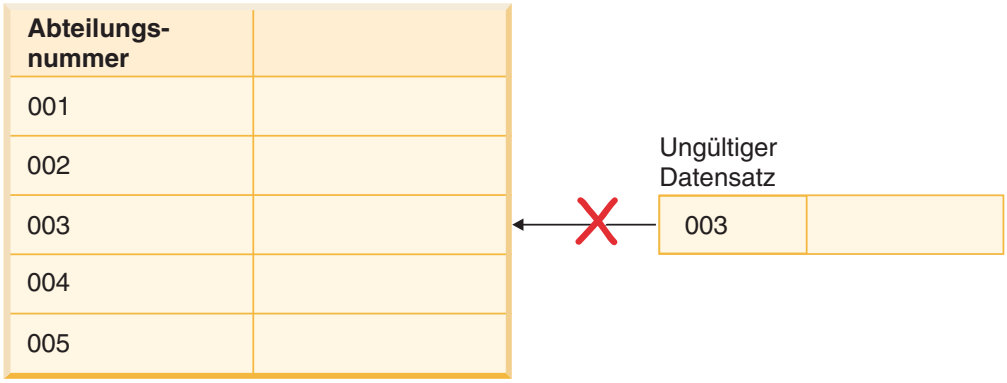


Abbildung 38. Eindeutige Integritätsbedingungen verhindern Datenduplikate

Der Datenbankmanager beachtet die Integritätsbedingung bei Operationen zum Einfügen und Aktualisieren von Daten, um die Datenintegrität zu gewährleisten.

Eine Tabelle kann eine beliebige Anzahl von Integritätsbedingungen für Eindeutigkeit haben, wobei maximal eine eindeutige Integritätsbedingung als Primärschlüssel definiert sein kann. Eine Tabelle kann nicht mehr als eine eindeutige Integritätsbedingung für dieselbe Gruppe von Spalten haben.

Eine eindeutige Integritätsbedingung, auf die im Fremdschlüssel einer referenziellen Integritätsbedingung verwiesen wird, heißt *übergeordneter Schlüssel*.

- Wenn eine eindeutige Integritätsbedingung in einer Anweisung CREATE TABLE definiert wird, wird vom Datenbankmanager automatisch ein eindeutiger Index erstellt und als systemerforderlicher Primärindex bzw. eindeutiger Index gekennzeichnet.
- Wenn eine eindeutige Integritätsbedingung in einer Anweisung ALTER TABLE definiert wird und ein Index für die gleichen Spalten vorhanden ist, wird dieser vorhandene Index als eindeutig und systemerforderlich gekennzeichnet. Falls kein solcher Index vorhanden ist, wird vom Datenbankmanager automatisch ein eindeutiger Index erstellt und als systemerforderlicher Primärindex bzw. eindeutiger Index gekennzeichnet.

Anmerkung: Es besteht ein Unterschied zwischen dem Definieren einer eindeutigen Integritätsbedingung und dem Erstellen eines eindeutigen Index. Zwar dienen beide zur Erhaltung der Eindeutigkeit, jedoch lässt ein eindeutiger Index Spalten mit möglichen Nullwerten zu und kann in der Regel nicht als übergeordneter Schlüssel verwendet werden.

Integritätsbedingungen über Primärschlüssel

Integritätsbedingungen über Primärschlüssel und Fremdschlüssel dienen zur Definition von Beziehungen zwischen Tabellen.

Ein Primärschlüssel ist eine Spalte oder eine Kombination von Spalten, die die gleichen Eigenschaften wie eine eindeutige Integritätsbedingung besitzen. Da ein Primärschlüssel zur Angabe einer Zeile in einer Tabelle verwendet wird, muss er eindeutig sein und über das Attribut NOT NULL verfügen. Eine Tabelle kann nicht mehr als einen Primärschlüssel, jedoch mehrere eindeutige Schlüssel besitzen. Primärschlüssel sind wahlfrei und können definiert werden, wenn eine Tabelle erstellt oder geändert wird. Sie besitzen zudem den weiteren Vorteil, dass sie für eine Reihenfolge der Daten sorgen, wenn Daten exportiert oder reorganisiert werden.

Prüfungen auf Integritätsbedingungen (in Tabellen)

Eine *Prüfung auf Integritätsbedingung* (auch als *Prüfung auf Integritätsbedingung in Tabellen* bezeichnet) ist eine Datenbankregel, mit der die zulässigen Werte in einer oder mehreren Spalten jeder Zeile einer Tabelle angegeben werden. Die Angabe von Prüfungen auf Integritätsbedingungen erfolgt durch ein eingeschränktes Format einer Suchbedingung.

Integritätsbedingungen über Fremdschlüssel (referenzielle Integritätsbedingungen)

Integritätsbedingungen über Fremdschlüssel (die auch als *referenzielle Integritätsbedingungen* bezeichnet werden) geben Ihnen die Möglichkeit, erforderliche Beziehungen zwischen und innerhalb von Tabellen zu definieren.

Zum Beispiel könnte eine typische Integritätsbedingung über Fremdschlüssel festlegen, dass jeder Mitarbeiter in der Tabelle EMPLOYEE ein Mitglied einer bestehenden, in der Tabelle DEPARTMENT definierten Abteilung sein muss.

Als *referenzielle Integrität* wird der Status einer Datenbank bezeichnet, in der alle Werte aller Fremdschlüssel gültig sind. Ein *Fremdschlüssel* ist eine Spalte oder eine Gruppe von Spalten in einer Tabelle, deren Werte mit mindestens einem Wert des Primärschlüssels oder eines eindeutigen Schlüssels einer Zeile in der übergeordneten Tabelle übereinstimmen müssen. Eine *referenzielle Integritätsbedingung* ist die Regel, dass die Werte des Fremdschlüssels nur dann gültig sind, wenn eine der folgenden Bedingungen gilt:

- Sie treten als Werte eines übergeordneten Schlüssels auf.
- Eine Komponente des Fremdschlüssels ist NULL.

Zur Herstellung dieser Beziehung würden Sie die Abteilungsnummer (WORKDEPT) der Tabelle EMPLOYEE als Fremdschlüssel und die Abteilungsnummer (DEPTNO) der Tabelle DEPARTMENT als Primärschlüssel definieren.

In Abb. 39 auf Seite 451 sehen Sie, wie ein Datensatz mit einem ungültigen Schlüssel nicht zu einer Tabelle hinzugefügt wird, wenn zwischen zwei Tabellen eine Integritätsbedingung über Fremdschlüssel vorhanden ist:

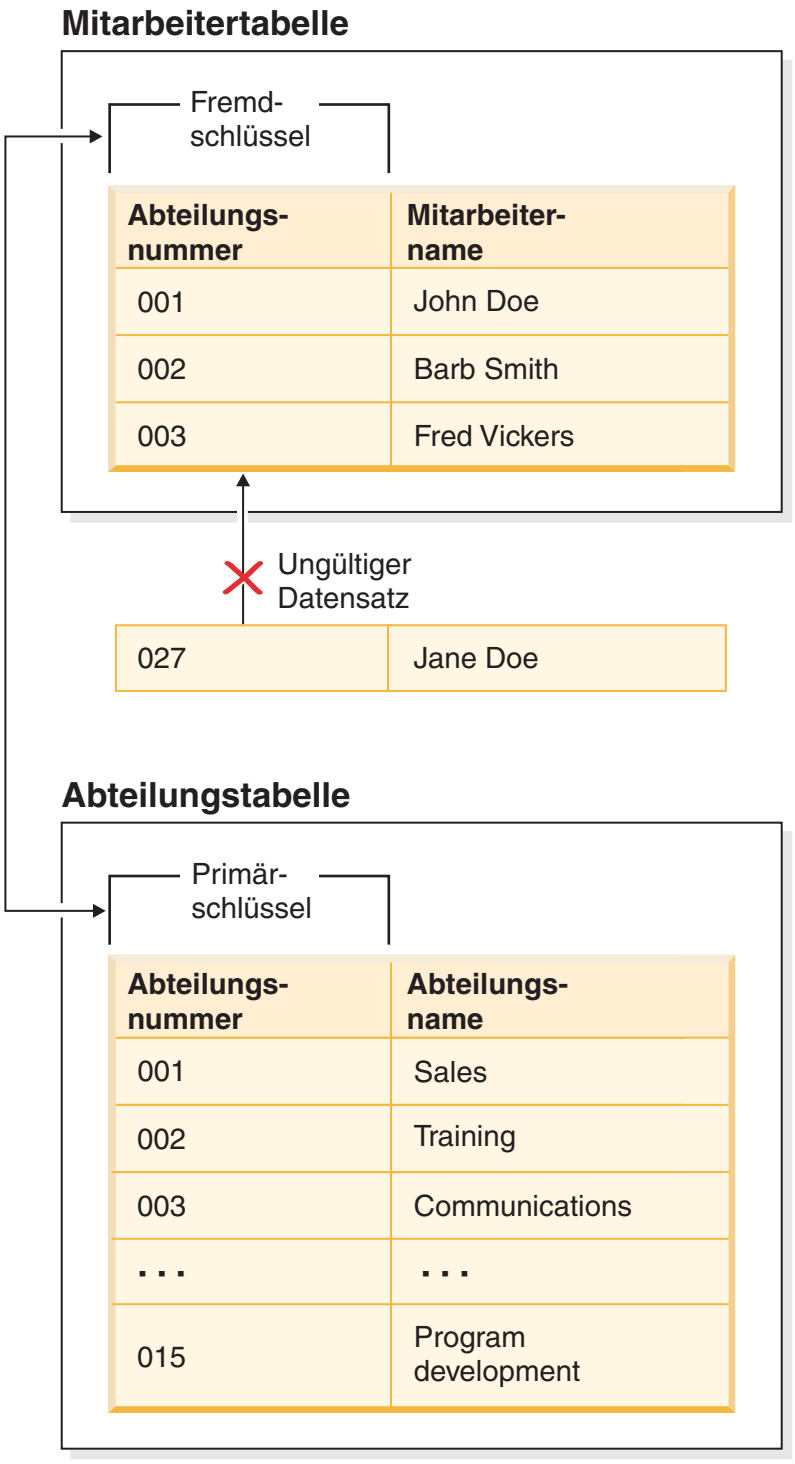


Abbildung 39. Integritätsbedingungen über Fremd- und Primärschlüssel

Die Tabelle, die den übergeordneten Schlüssel enthält, wird als *übergeordnete Tabelle* der referenziellen Integritätsbedingung bezeichnet, während die Tabelle, die den Fremdschlüssel enthält, als *abhängige Tabelle* dieser Tabelle bezeichnet wird.

Referenzielle Integritätsbedingungen können in der Anweisung CREATE TABLE oder ALTER TABLE definiert werden. Referenzielle Integritätsbedingungen werden

durch den Datenbankmanager bei der Ausführung der Anweisungen INSERT, UPDATE, DELETE, ALTER TABLE, MERGE, ADD CONSTRAINT und SET INTEGRITY umgesetzt.

Die Regeln der referenziellen Integrität lassen sich durch die folgenden Termini definieren:

Tabelle 76. Termini zur referenziellen Integrität

Konzept	**Termini**
Übergeordneter Schlüssel	Ein Primärschlüssel bzw. ein eindeutiger Schlüssel einer referenziellen Integritätsbedingung.
Übergeordnete Zeile	Eine Zeile, die mindestens eine von ihr abhängige Zeile hat.
Übergeordnete Tabelle	Eine Tabelle, die den übergeordneten Schlüssel einer referenziellen Integritätsbedingung enthält. Eine Tabelle kann für beliebig viele referenzielle Integritätsbedingungen die übergeordnete Tabelle sein. Eine Tabelle, die in einer referenziellen Integritätsbedingung übergeordnete Tabelle ist, kann gleichzeitig abhängige Tabelle in einer referenziellen Integritätsbedingung sein.
Abhängige Tabelle	Eine Tabelle, die in ihrer Definition mindestens eine referenzielle Integritätsbedingung enthält. Eine Tabelle kann für beliebig viele referenzielle Integritätsbedingungen eine abhängige Tabelle sein. Eine Tabelle, die in einer referenziellen Integritätsbedingung eine abhängige Tabelle ist, kann gleichzeitig die übergeordnete Tabelle in einer referenziellen Integritätsbedingung sein.
Untergeordnete Tabelle	Eine Tabelle ist eine untergeordnete Tabelle von Tabelle T, wenn sie eine abhängige Tabelle von T oder eine untergeordnete Tabelle einer abhängigen Tabelle von T ist.
Abhängige Zeile	Eine Zeile, die mindestens eine übergeordnete Zeile hat.
Untergeordnete Zeile	Eine Zeile ist eine untergeordnete Zeile von Zeile Z, wenn sie eine abhängige Zeile von Z oder eine untergeordnete Zeile einer abhängigen Zeile von Z ist.
Referenzieller Zyklus	Eine Gruppe referenzieller Integritätsbedingungen, in der jede Tabelle in der Gruppe eine untergeordnete Tabelle zu sich selbst ist.
Auf sich selbst verweisende Tabelle	Eine Tabelle, die in der gleichen referenziellen Integritätsbedingung sowohl übergeordnete als auch abhängige Tabelle ist. Die entsprechende Integritätsbedingung wird als *auf sich selbst verweisende Integritätsbedingung* bezeichnet.
Auf sich selbst verweisende Zeile	Eine Zeile, die eine übergeordnete Zeile zu sich selbst ist.

Der Zweck einer referenziellen Integritätsbedingung ist, zu garantieren, dass Tabellenbeziehungen gepflegt und Dateneingaberegeln befolgt werden. Dies bedeutet Folgendes: Solange eine referenzielle Integritätsbedingung gültig ist, garantiert der Datenbankmanager, dass für jede Zeile in einer untergeordneten Tabelle, die über einen Wert ungleich null in ihren Fremdschlüsselspalten verfügt, eine Zeile in einer entsprechenden übergeordneten Tabelle vorhanden ist, die über einen übereinstimmenden Wert in ihrem Primärschlüssel verfügt.

Wenn bei einer SQL-Operation versucht wird, Daten so zu ändern, dass die referenzielle Integrität beeinträchtigt wird, könnte gegen eine Integritätsbedingung über Fremdschlüssel (referenzielle Integritätsbedingung) verstoßen werden. Der Datenbankmanager meistert derartige Situationen, indem er eine Gruppe von Re-

geln umsetzt, die den einzelnen referenziellen Integritätsbedingungen zugeordnet sind. Diese Gruppe von Regeln besteht aus folgenden Regeln:

- Einer Einfügeregel
- Einer Aktualisierungsregel
- Einer Löschregel

Wenn bei einer SQL-Operation versucht wird, Daten so zu ändern, dass die referenzielle Integrität beeinträchtigt wird, könnte gegen eine referenzielle Integritätsbedingung verstoßen werden. Beispiel:

- Eine Einfügeoperation könnte versuchen, einer untergeordneten Tabelle, die in ihren Fremdschlüsselspalten über einen Wert verfügt, der keinem Wert im Primärschlüssel der entsprechenden übergeordneten Tabelle entspricht, eine Zeile mit Daten hinzuzufügen.
- Eine Aktualisierungsoperation könnte versuchen, den Wert in den Fremdschlüsselspalten einer untergeordneten Tabelle in einen Wert zu ändern, der über keinen übereinstimmenden Wert im Primärschlüssel der entsprechenden übergeordneten Tabelle verfügt.
- Eine Aktualisierungsoperation könnte versuchen, den Wert im Primärschlüssel einer übergeordneten Tabelle in einen Wert zu ändern, der über keinen übereinstimmenden Wert in den Fremdschlüsselspalten einer untergeordneten Tabelle verfügt.
- Eine Löschoperation könnte versuchen, einen Datensatz aus einer übergeordneten Tabelle, die über einen übereinstimmenden Wert in den Fremdschlüsseltabellen einer untergeordneten Tabelle verfügt, zu entfernen.

Der Datenbankmanager meistert derartige Situationen, indem er eine Gruppe von Regeln umsetzt, die den einzelnen referenziellen Integritätsbedingungen zugeordnet sind. Diese Gruppe von Regeln besteht aus folgenden Regeln:

- Einer Einfügeregel
- Einer Aktualisierungsregel
- Einer Löschregel

Einfügeregel

Die Einfügeregel einer referenziellen Integritätsbedingung legt fest, dass ein einzufügender Nicht-Nullwert des Fremdschlüssels mit einem Wert des übergeordneten Schlüssels der übergeordneten Tabelle übereinstimmen muss. Der Wert eines zusammengesetzten Fremdschlüssels ist NULL, wenn eine Komponente des Werts NULL ist. Diese Regel gilt implizit, wenn ein Fremdschlüssel definiert wird.

Aktualisierungsregel

Die Aktualisierungsregel einer referenziellen Integritätsbedingung wird bei der Definition der referenziellen Integritätsbedingung angegeben. Angegeben werden kann NO ACTION oder RESTRICT. Die Aktualisierungsregel wird angewendet, wenn eine Zeile der übergeordneten Tabelle oder eine Zeile der abhängigen Tabelle aktualisiert wird.

Für eine übergeordnete Zeile gelten folgende Regeln, wenn ein Wert in einer Spalte des übergeordneten Schlüssels aktualisiert wird:

- Wenn irgendeine Zeile in der abhängigen Tabelle mit dem Ursprungswert des Schlüssels übereinstimmt, wird die UPDATE-Operation zurückgewiesen, wenn als Aktualisierungsregel RESTRICT definiert ist.

- Wenn irgendeine Zeile der abhängigen Tabelle nach Ausführung der UPDATE-Anweisung (ausgenommen AFTER-Trigger) keinen entsprechenden übergeordneten Schlüssel hat, wird die Aktualisierung zurückgewiesen, wenn als Aktualisierungsregel NO ACTION definiert ist.

Der Wert der übergeordneten eindeutigen Schlüssel kann nicht geändert werden, wenn es sich bei der Aktualisierungsregel um RESTRICT handelt und mindestens eine abhängige Zeile vorhanden ist. Handelt es sich jedoch um die Aktualisierungsregel NO ACTION, können übergeordnete eindeutige Schlüssel aktualisiert werden, so lange wie jedes untergeordnete Element über einen übergeordneten Schlüssel verfügt, wenn die Aktualisierungsanweisung beendet wird. Ein Aktualisierungswert eines Fremdschlüssels, der nicht NULL ist, muss mit einem Wert des Primärschlüssels aus der übergeordneten Tabelle der Beziehung identisch sein.

Mit der Verwendung von NO ACTION oder RESTRICT als Aktualisierungsregeln für referenzielle Integritätsbedingungen wird darüber hinaus festgestellt, wann die Integritätsbedingung erzwungen wird. Eine Aktualisierungsregel RESTRICT wird vor allen anderen Integritätsbedingungen, einschließlich der referenziellen Integritätsbedingungen mit Modifizierungsregeln, wie z. B. CASCADE oder SET NULL, erzwungen. Eine Aktualisierungsregel NO ACTION wird nach anderen referenziellen Integritätsbedingungen erzwungen. Beachten Sie, dass der zurückgegebene SQLSTATE-Wert sich unterscheidet, je nachdem ob die Aktualisierungsregel RESTRICT oder NO ACTION lautet.

Für eine abhängige Zeile gilt die Aktualisierungsregel NO ACTION implizit, wenn ein Fremdschlüssel angegeben wird. NO ACTION bedeutet, dass ein Aktualisierungswert eines Fremdschlüssels, der nicht NULL ist, mit einem Wert des übergeordneten Schlüssels der übergeordneten Tabelle übereinstimmen muss, wenn die UPDATE-Anweisung ausgeführt ist.

Der Wert eines zusammengesetzten Fremdschlüssels ist NULL, wenn eine Komponente des Werts NULL ist.

Löschregel

Die Löschregel einer referenziellen Integritätsbedingung wird bei der Definition der referenziellen Integritätsbedingung angegeben. Angegeben werden kann NO ACTION, RESTRICT, CASCADE oder SET NULL. SET NULL kann nur angegeben werden, wenn eine Spalte des Fremdschlüssels Nullwerte zulässt.

Wenn die angegebene Tabelle oder die Basistabelle der angegebenen Sicht eine übergeordnete Tabelle ist, dürfen die zum Löschen ausgewählten Zeilen keine abhängigen Zeilen in einer Beziehung mit einer Löschregel RESTRICT aufweisen und DELETE darf nicht an die untergeordneten Zeilen weitergegeben werden, die abhängige Zeilen in einer Beziehung mit einer Löschregel RESTRICT aufweisen.

Wenn die Löschoperation nicht von einer Löschregel RESTRICT verhindert wird, werden die ausgewählten Zeilen gelöscht. Jede Zeile, die von den ausgewählten Zeilen abhängig ist, ist ebenso betroffen:

- Die Spalten, die Nullwerte enthalten dürfen, der Fremdschlüssel von beliebigen Zeilen, die ihre abhängigen Zeilen in einer Beziehung mit einer Löschregel SET NULL darstellen, werden auf den Nullwert gesetzt.
- Alle Zeilen, die ihre abhängigen Zeilen in einer Beziehung mit einer Löschregel CASCADE darstellen, werden ebenfalls gelöscht und die oben genannten Regeln gelten ihrerseits für diese Zeilen.

Die Löschregel NO ACTION wurde aktiviert, um zu erzwingen, dass jeder Fremdschlüssel, der einen Wert ungleich null aufweist, auf eine vorhandene übergeordnete Zeile verweist, nachdem die anderen referenziellen Integritätsbedingungen erzwungen wurden.

Die Löschregel einer referenziellen Integritätsbedingung wird nur angewendet, wenn *eine Zeile* der übergeordneten Tabelle gelöscht wird. Genauer gesagt, die Regel wird nur angewendet, wenn *eine Zeile* der übergeordneten Tabelle das Objekt einer Löschoperation bzw. einer weitergegebenen Löschoperation (im folgenden Abschnitt definiert) ist und die Zeile abhängige Zeilen in der abhängigen Tabelle der referenziellen Integritätsbedingung hat. Betrachten Sie ein Beispiel, in dem P die übergeordnete Tabelle, D die abhängige Tabelle und p die übergeordnete Zeile sind, die das Objekt einer Löschoperation bzw. einer weitergegebenen Löschoperation ist. Die Löschregel funktioniert wie folgt:

- Bei RESTRICT oder NO ACTION tritt ein Fehler auf und keine Zeilen werden gelöscht.
- Bei CASCADE wird die Löschoperation an die abhängigen Zeilen von p in Tabelle D weitergegeben.
- Bei SET NULL wird jede Spalte mit optionaler Dateneingabe (NULLABLE) des Fremdschlüssels jeder abhängigen Zeile von p in Tabelle D auf den Wert NULL gesetzt.

Jede Tabelle, die an einer Löschoperation von Tabelle P beteiligt sein kann, wird als durch *übergreifendes Löschen* mit P verbunden bezeichnet. Das heißt, eine Tabelle ist mit Tabelle P durch übergreifendes Löschen verbunden, wenn sie eine abhängige Tabelle von P oder einer Tabelle ist, an die Löschoperationen von P weitergegeben werden.

Die folgenden Einschränkungen gelten für die Abhängigkeiten zwischen Tabellen, die durch übergreifendes Löschen miteinander verbunden sind:

- Wenn eine Tabelle durch übergreifendes Löschen in einem referenziellen Zyklus von mehr als einer Tabelle mit sich selbst verbunden ist, darf der Zyklus keine Löschregel RESTRICT oder SET NULL enthalten.
- Eine Tabelle darf nicht gleichzeitig eine abhängige Tabelle in einer CASCADE-Abhängigkeit (auf sich selbst oder auf eine andere Tabelle verweisend) sein und eine auf sich selbst verweisende Beziehung mit einer Löschregel RESTRICT oder SET NULL haben.
- Wenn eine Tabelle durch übergreifendes Löschen mit einer anderen Tabelle über mehrere Abhängigkeiten verbunden ist, wobei diese Abhängigkeiten überlappende Fremdschlüssel haben, müssen diese Abhängigkeiten die gleiche Löschregel haben und keine dieser Löschregeln kann SET NULL sein.
- Wenn eine Tabelle durch übergreifendes Löschen mit einer anderen Tabelle über mehrere Abhängigkeiten verbunden ist, wobei eine der Beziehungen mit der Löschregel SET NULL angegeben ist, darf die Fremdschlüsseldefinition dieser Abhängigkeit keine Verteilungsschlüssel- oder MDC-Schlüsselspalte, keine Tabellenpartitionierungsschlüsselspalte oder RCT-Schlüsselspalte enthalten.
- Wenn zwei Tabellen durch übergreifendes Löschen mit derselben Tabelle über CASCADE-Abhängigkeiten verbunden sind, dürfen die beiden Tabellen nicht durch ein übergreifendes Löschen miteinander verbunden werden, bei dem die Pfade der durch übergreifendes Löschen definierten Abhängigkeiten mit der Löschregel RESTRICT oder SET NULL enden.

Informative Integritätsbedingungen

Eine *informative Integritätsbedingung* ist ein Integritätsbedingungsattribut, das vom SQL-Compiler zur Verbesserung des Zugriffs auf Daten verwendet werden kann. Informative Integritätsbedingungen werden nicht durch den Datenbankmanager umgesetzt und dienen nicht zur weiteren Überprüfung von Daten, sondern zur Verbesserung der Abfrageleistung.

Informative Integritätsbedingungen werden mithilfe der Anweisungen CREATE TABLE oder ALTER TABLE definiert. Sie fügen zuerst referenzielle Integritätsbedingungen oder Prüfungen auf Integritätsbedingungen hinzu und ordnen diesen dann Integritätsbedingungsattribute zu, indem Sie angeben, ob der Datenbankmanager die Integritätsbedingung umsetzen soll oder nicht. Für Prüfungen auf Integritätsbedingungen können Sie außerdem angeben, dass die Integritätsbedingung anerkannt ist. Für referenzielle Integritätsbedingungen gilt: Wenn die Integritätsbedingung nicht umgesetzt wird, können Sie zusätzlich angeben, ob die Integritätsbedingung anerkannt ist oder nicht. Eine nicht umgesetzte und nicht anerkannte Integritätsbedingung wird auch als statistische referenzielle Integritätsbedingung bezeichnet. Nach dem Angeben der Integritätsbedingung können Sie angeben, ob die Integritätsbedingung zur Abfrageoptimierung verwendet werden soll.

Entwerfen von Integritätsbedingungen

Beim Entwerfen und Erstellen von Integritätsbedingungen empfiehlt es sich, eine Namenskonvention zu verwenden, die eine geeignete Unterscheidung der verschiedenen Typen von Integritätsbedingungen ermöglicht. Dies spielt insbesondere für die Diagnose von Fehlern, die möglicherweise auftreten, eine wichtige Rolle.

Informationen zu diesem Vorgang

Sie können die folgenden Integritätsbedingungstypen entwerfen:

- NOT NULL, Integritätsbedingung
- Eindeutige Integritätsbedingungen
- Integritätsbedingungen über Primärschlüssel
- Prüfungen auf Integritätsbedingungen (in Tabellen)
- Integritätsbedingungen über Fremdschlüssel (referenzielle Integritätsbedingungen)
- Informative Integritätsbedingungen

Entwerfen eindeutiger Integritätsbedingungen

Eindeutige Integritätsbedingungen stellen sicher, dass jeder Wert im angegebenen Schlüssel eindeutig ist. Eine Tabelle kann über mehrere eindeutige Integritätsbedingungen verfügen, wobei eine eindeutige Integritätsbedingung als Primärschlüssel definiert sein kann.

Informationen zu diesem Vorgang

Eine eindeutige Integritätsbedingung wird mit der Klausel UNIQUE in der Anweisung CREATE TABLE bzw. ALTER TABLE definiert. Ein eindeutiger Schlüssel kann aus mehr als einer Spalte bestehen. In einer Tabelle ist mehr als eine eindeutige Integritätsbedingung zulässig.

Wenn die eindeutige Integritätsbedingung definiert ist, wird sie vom Datenbankmanager automatisch umgesetzt, wenn eine INSERT- oder UPDATE-Anweisung die Daten in der Tabelle ändert. Die eindeutige Integritätsbedingung wird mithilfe eines eindeutigen Index realisiert.

Wenn eine eindeutige Integritätsbedingung in einer Anweisung ALTER TABLE definiert wird und ein Index für dieselbe Gruppe von Spalten dieses eindeutigen Schlüssels existiert, wird dieser Index zum eindeutigen Index und wird von der Integritätsbedingung verwendet.

Sie können jede einzelne eindeutige Integritätsbedingung als Primärschlüssel verwenden. Der Primärschlüssel kann als übergeordneter Schlüssel in einer referenziellen Integritätsbedingung (zusammen mit anderen eindeutigen Integritätsbedingungen) verwendet werden. Ein Primärschlüssel wird mit der Klausel PRIMARY KEY in der Anweisung CREATE TABLE bzw. ALTER TABLE definiert. Der Primärschlüssel kann aus mehr als einer Spalte bestehen.

Der Primärindex sorgt zwingend dafür, dass der Primärschlüssel eindeutig ist. Wenn eine Tabelle mit einem Primärschlüssel erstellt wird, legt der Datenbankmanager einen Primärindex für diesen Schlüssel an.

Einige Hinweise zur Leistungsoptimierung für Indizes, die für eindeutige Integritätsbedingungen verwendet werden:

- Beim einleitenden Laden einer leeren Tabelle mit Indizes können mit **LOAD** bessere Leistungen erzielt werden als mit **IMPORT**. Dies ist von der Verwendung des Modus **INSERT** oder **REPLACE** für **LOAD** unabhängig.
- Beim Anhängen einer beträchtlichen Datenmenge an eine vorhandene Tabelle mit Indizes (unter Verwendung von **IMPORT INSERT** oder **LOAD INSERT**) ist **LOAD** etwas leistungsstärker als **IMPORT**.
- Wenn Sie den Befehl **IMPORT** für das einleitende Laden einer großen Datenmenge verwenden, erstellen Sie den eindeutigen Schlüssel nach dem Import bzw. Laden der Daten. Dadurch erübrigt sich die Verwaltung des Index beim Laden der Tabelle. Darüber hinaus verwendet der Index auf diese Weise die geringste Menge an Speicher.
- Wenn Sie das Dienstprogramm zum Laden (**LOAD**) im Modus **REPLACE** verwenden, erstellen Sie den eindeutigen Schlüssel, bevor Sie die Daten laden. In diesem Fall ist die Erstellung des Index während des Ladens effizienter als die Verwendung der Anweisung CREATE INDEX nach der Ladeoperation.

Einschränkungen

- Eine eindeutige Integritätsbedingung kann nicht in einer untergeordneten Tabelle definiert werden.
- Es kann nur einen Primärschlüssel pro Tabelle geben.

Entwerfen von Integritätsbedingungen über Primärschlüssel

Jede Tabelle kann einen und nur einen Primärschlüssel besitzen. Ein Primärschlüssel ist eine Spalte oder eine Kombination von Spalten, die die gleichen Eigenschaften wie eine eindeutige Integritätsbedingung besitzen. Ein Primärschlüssel und Integritätsbedingungen über Fremdschlüssel dienen zur Definition von Beziehungen zwischen Tabellen.

Informationen zu diesem Vorgang

Da ein Primärschlüssel zur Angabe einer Zeile in einer Tabelle verwendet wird, muss er eindeutig sein und möglichst wenigen Hinzufüge- oder Löschoperationen unterliegen. Eine Tabelle kann nicht mehr als einen Primärschlüssel, jedoch mehrere eindeutige Schlüssel besitzen. Primärschlüssel sind optional und können mit der Klausel PRIMARY KEY definiert werden, wenn eine Tabelle erstellt oder geändert wird. Sie besitzen zudem den weiteren Vorteil, dass sie für eine Reihenfolge der Daten sorgen, wenn Daten exportiert oder reorganisiert werden.

Integritätsbedingung über Primärschlüssel werden wie eindeutige Integritätsbedingungen entworfen (siehe „Entwerfen eindeutiger Integritätsbedingungen" auf Seite 456). Der einzige Unterschied besteht darin, dass im Gegensatz zu eindeutigen Integritätsbedingungen jeweils nur ein Primärschlüssel pro Tabelle definiert werden kann.

Anmerkung: Integritätsbedingungen über Primärschlüssel, die auf zusammengesetzten Primärschlüsseln basieren, sind zulässig.

Entwerfen von Prüfungen auf Integritätsbedingungen

Bei der Erstellung von Prüfungen auf Integritätsbedingungen können zwei Fälle auftreten: (i) Alle Zeilen entsprechen der Prüfung auf Integritätsbedingung. (ii) Einige oder alle Zeilen entsprechen der Prüfung auf Integritätsbedingung nicht.

Informationen zu diesem Vorgang

Alle Zeilen entsprechen der Prüfung auf Integritätsbedingung
Wenn alle Zeilen der Prüfung auf Integritätsbedingung entsprechen, wird die Prüfung auf Integritätsbedingung erfolgreich erstellt. Nachfolgende Versuche, Daten einzufügen oder zu aktualisieren, die der Integritätsgeschäftsregel nicht entsprechen, werden zurückgewiesen.

Einige oder alle Zeilen entsprechen der Prüfung auf Integritätsbedingung nicht
Wenn einige Zeilen vorhanden sind, die der Prüfung auf Integritätsbedingung nicht entsprechen, wird die Prüfung auf Integritätsbedingung nicht erstellt (d. h. die Ausführung der Anweisung ALTER TABLE schlägt fehl). Das folgende Beispiel zeigt eine Anweisung ALTER TABLE, die einer Tabelle EMPLOYEE eine neue Integritätsbedingung hinzufügt. Die Prüfung auf Integritätsbedingung hat den Namen CHECK_JOB. Der Datenbankmanager verwendet diesen Namen, um Sie darüber zu informieren, gegen welche Integritätsbedingung verstoßen wurde, wenn eine INSERT- oder UPDATE-Anweisung fehlschlägt. Die Klausel CHECK wird zur Definition einer Prüfung auf Integritätsbedingung in einer Tabelle verwendet.

```
ALTER TABLE EMPLOYEE
  ADD CONSTRAINT check_job
  CHECK (JOB IN ('Engineer', 'Sales', 'Manager'));
```

Es wurde eine Anweisung ALTER TABLE verwendet, da die Tabelle bereits definiert war. Falls in der Tabelle EMPLOYEE Werte vorhanden sind, die gegen die zu definierende Integritätsbedingung verstoßen, wird die Anweisung ALTER TABLE nicht erfolgreich ausgeführt.

Da Prüfungen auf Integritätsbedingungen und andere Typen von Integritätsbedingungen zur Implementierung von Geschäftsregeln dienen, müssen sie möglicherweise ab und zu geändert werden. Dies kann zum Beispiel der Fall sein, wenn sich Geschäftsregeln in einem Unternehmen ändern. Wenn eine Prüfung auf Integritätsbedingung geändert werden muss, kann dies nur dadurch geschehen, dass sie ge-

löscht und eine neue Prüfung auf Integritätsbedingung erstellt wird. Prüfungen auf Integritätsbedingungen können jederzeit gelöscht werden, und die Tabelle oder die darin enthaltenen Daten werden von dieser Aktion nicht berührt. Wenn Sie eine Prüfung auf Integritätsbedingung löschen, müssen Sie beachten, dass die Datenprüfung, die durch die Integritätsbedingung ausgeführt wurde, nicht mehr stattfindet.

Vergleich zwischen Prüfungen auf Integritätsbedingungen und BEFORE-Triggern

Sie müssen den Unterschied zwischen Prüfungen auf Integritätsbedingungen und Triggern berücksichtigen, wenn Sie die Verwendung von Triggern oder Prüfungen auf Integritätsbedingungen zur Wahrung der Integrität Ihrer Daten in Betracht ziehen.

In einer relationalen Datenbank muss die Datenintegrität gewährleistet werden, wenn mehrere Benutzer auf die Daten zugreifen oder sie ändern. Immer dann, wenn Daten gemeinsam verwendet werden, muss die Richtigkeit der Werte innerhalb von Datenbanken sichergestellt werden.

Prüfungen auf Integritätsbedingungen

Eine Prüfung auf Integritätsbedingung (in Tabellen) definiert Einschränkungen für Daten, die einer bestimmten Tabelle hinzugefügt werden. Mithilfe einer Prüfung auf Integritätsbedingung in Tabellen können Sie Einschränkungen über die Datentypeinschränkungen hinaus für die Werte definieren, die für eine Spalte in der Tabelle zulässig sind. Prüfungen auf Integritätsbedingungen in Tabellen werden in Form von Bereichsprüfungen oder Prüfungen auf andere Werte in derselben Zeile derselben Tabelle durchgeführt.

Wenn die Regel für alle Anwendungen gilt, die die Daten verwenden, müssen Sie eine Prüfung auf Integritätsbedingung in Tabellen verwenden, um Ihre Einschränkung für die in der Tabelle zulässigen Daten umzusetzen. Mithilfe von Prüfungen auf Integritätsbedingungen in Tabellen kann die Einschränkung allgemein angewendet und kann einfacher verwaltet werden.

Die Umsetzung der Prüfungen auf Integritätsbedingungen ist für die Pflege der Datenintegrität wichtig. Jedoch wird dabei auch eine bestimmte Systemaktivität ausgelöst, die die Leistung beeinflussen kann, sobald große Datenvolumen geändert werden.

BEFORE-Trigger

Durch die Verwendung von Triggern, die vor einer Aktualisierungs- oder Einfügeoperation ausgeführt werden, können Werte, die aktualisiert oder eingefügt wurden, geändert werden, *bevor* die Datenbank tatsächlich geändert wird. Falls dies gewünscht wird, können sie zur Umsetzung von Eingaben aus dem Anwendungsformat (Benutzersicht der Daten) in ein internes Datenbankformat verwendet werden. BEFORE-Trigger können auch zur Aktivierung anderer Operationen als Datenbankoperationen über benutzerdefinierte Funktionen verwendet werden.

Neben der Änderung ist die Datenprüfung ein häufiges Einsatzgebiet von BEFORE-Triggern, wobei die Klausel SIGNAL verwendet wird.

Zwischen BEFORE-Triggern und Prüfungen auf Integritätsbedingungen bestehen zwei Unterschiede, wenn sie für die Datenprüfung verwendet werden:

1. BEFORE-Trigger unterliegen im Gegensatz zu Prüfungen auf Integritätsbedingungen keiner Einschränkung hinsichtlich des Zugriffs auf andere Werte in derselben Zeile derselben Tabelle.
2. Bei einer SET INTEGRITY-Operation für eine Tabelle nach einer LOAD-Operation werden Trigger (einschließlich BEFORE-Trigger) nicht ausgeführt. Prüfungen auf Integritätsbedingungen werden jedoch ausgeführt.

Entwerfen von (referenziellen) Integritätsbedingungen über Fremdschlüssel

Referenzielle Integrität wird implementiert, indem den Tabellen- und Spaltendefinitionen Integritätsbedingungen über Fremdschlüssel (oder referenzielle Integritätsbedingungen) hinzugefügt werden und um einen Index für alle Fremdschlüsselspalten zu erstellen. Wenn der Index und die Integritätsbedingungen über Fremdschlüssel definiert sind, werden Änderungen an Daten in den Tabellen und Spalten an der definierten Integritätsbedingung überprüft. Die Durchführung der angeforderten Aktion hängt von dem Ergebnis der Prüfung der Integritätsbedingung ab.

Informationen zu diesem Vorgang

Referenzielle Integritätsbedingungen werden mit der Klausel FOREIGN KEY und der Klausel REFERENCES in der Anweisung CREATE TABLE oder ALTER TABLE definiert. Es gibt Auswirkungen einer referenziellen Integritätsbedingung auf eine typisierte Tabelle bzw. auf eine übergeordnete Tabelle, die eine typisierte Tabelle ist, die Sie vor der Erstellung einer referenziellen Integritätsbedingung berücksichtigen sollten.

Die Definition von Fremdschlüsseln implementiert Integritätsbedingungen für die Werte innerhalb der Zeilen einer Tabelle oder zwischen den Zeilen zweier Tabellen. Der Datenbankmanager prüft die in einer Tabellendefinition angegebenen Integritätsbedingungen und verwaltet die Abhängigkeitsbeziehungen entsprechend. Das Ziel besteht darin, die Integrität ohne Leistungseinbußen zu erhalten, wenn ein Datenbankobjekt auf ein anderes verweist.

Zum Beispiel enthalten sowohl der Primärschlüssel als auch der Fremdschlüssel eine Spalte für die Abteilungsnummer. In der Tabelle EMPLOYEE heißt der Spaltenname WORKDEPT, während in der Tabelle DEPARTMENT der Name DEPTNO ist. Die Abhängigkeitsbeziehung zwischen diesen beiden Tabellen wird durch die folgenden Integritätsbedingungen definiert:

- Für jeden Mitarbeiter in der Tabelle EMPLOYEE gibt es nur eine Abteilungsnummer, und diese Nummer ist in der Tabelle DEPARTMENT vorhanden.
- Jede Zeile in der Tabelle EMPLOYEE verweist auf nur eine Zeile in der Tabelle DEPARTMENT. Es gibt eine eindeutige Beziehung zwischen den Tabellen.
- Jede Zeile in der Tabelle EMPLOYEE, die einen Nichtnullwert in der Spalte WORKDEPT enthält, hat eine Beziehung zu einer Zeile in der Spalte DEPTNO der Tabelle DEPARTMENT.
- Die Tabelle DEPARTMENT ist eine übergeordnete Tabelle, und die Tabelle EMPLOYEE ist die abhängige Tabelle.

Die Anweisung zur Definition der übergeordneten Tabelle DEPARTMENT sieht folgendermaßen aus:

```
CREATE TABLE DEPARTMENT
       (DEPTNO    CHAR(3)     NOT NULL,
        DEPTNAME  VARCHAR(29) NOT NULL,
```

```
        MGRNO     CHAR(6),
        ADMRDEPT  CHAR(3)     NOT NULL,
        LOCATION  CHAR(16),
           PRIMARY KEY (DEPTNO))
    IN RESOURCE
```

Die Anweisung zur Definition der abhängigen Tabelle EMPLOYEE sieht folgendermaßen aus:

```
CREATE TABLE EMPLOYEE
       (EMPNO     CHAR(6)     NOT NULL PRIMARY KEY,
        FIRSTNME  VARCHAR(12) NOT NULL,
        LASTNAME  VARCHAR(15) NOT NULL,
        WORKDEPT  CHAR(3),
        PHONENO   CHAR(4),
        PHOTO     BLOB(10m)   NOT NULL,
           FOREIGN KEY DEPT (WORKDEPT)
           REFERENCES DEPARTMENT ON DELETE NO ACTION)
    IN RESOURCE
```

Durch Angeben der Spalte DEPTNO als Primärschlüssel der Tabelle DEPARTMENT und der Spalte WORKDEPT als Fremdschüssel der Tabelle EMPLOYEE definieren Sie eine referenzielle Integritätsbedingung für die Werte der Spalte WORKDEPT. Durch diese Integritätsbedingung wird die referenzielle Integrität zwischen den Werten der beiden Tabellen implementiert. In vorliegenden Fall müssen alle Mitarbeiter (Employees), die der Tabelle EMPLOYEE hinzugefügt werden, eine Abteilungsnummer erhalten, die in der Tabelle DEPARTMENT enthalten ist.

Die Löschbedingung für die referenzielle Integritätsbedingung in der Tabelle EMPLOYEE ist NO ACTION. Das heißt, dass eine Abteilung (Department) nicht aus der Tabelle DEPARTMENT gelöscht werden kann, wenn es Mitarbeiter in dieser Abteilung gibt.

Zum Hinzufügen einer referenziellen Integritätsbedingung kann nicht nur die in den Beispielen gezeigte Anweisung CREATE TABLE, sondern auch die Anweisung ALTER TABLE verwendet werden.

Ein weiteres Beispiel: Es werden dieselben Tabellendefinitionen wie im vorigen Beispiel verwendet. Außerdem wird die Tabelle DEPARTMENT vor der Tabelle EMPLOYEE erstellt. Jede Abteilung hat einen Manager, und dieser Manager wird in der Tabelle EMPLOYEE aufgeführt. Die Spalte MGRNO der Tabelle DEPARTMENT stellt tatsächlich einen Fremdschlüssel der Tabelle EMPLOYEE dar. Aufgrund dieses referenziellen Zyklus führt diese Integritätsbedingung zu einem kleinen Problem. Sie könnten einen Fremdschlüssel später hinzufügen. Sie könnten außerdem die Anweisung CREATE SCHEMA verwenden, um die Tabellen EMPLOYEE und DEPARTMENT gleichzeitig zu erstellen.

Siehe auch „Fremdschlüssel in referenziellen Integritätsbedingungen" auf Seite 464.

Beispiele für die Interaktion zwischen Triggern und referenziellen Integritätsbedingungen

Aktualisierungsoperationen (UPDATE) können zu Interaktionen von Triggern mit referenziellen Integritätsbedingungen und Prüfungen auf Integritätsbedingungen führen.

Abb. 40 auf Seite 462 und die zugehörige Beschreibung sind für die Verarbeitung repräsentativ, die für eine Anweisung ausgeführt wird, die Daten in der Datenbank aktualisiert.

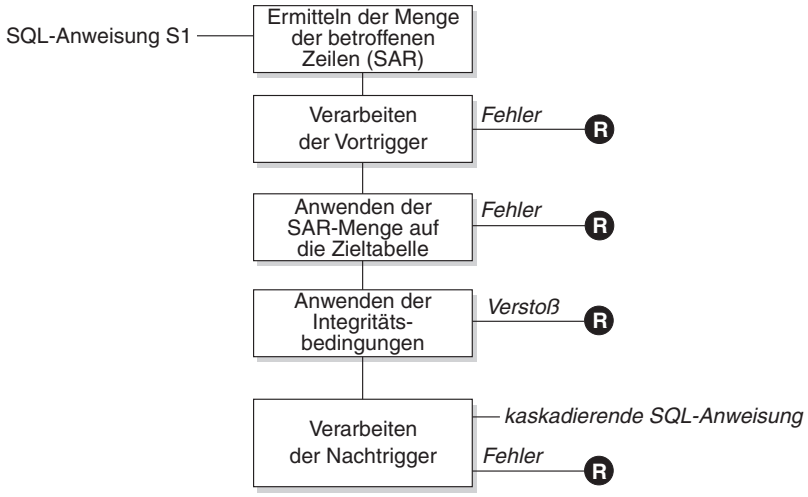


Abbildung 40. Verarbeitung einer Anweisung mit definierten Triggern und Integritätsbedingungen

Abb. 40 zeigt den allgemeinen Verarbeitungsablauf für eine Anweisung, die eine Tabelle aktualisiert. Sie geht von einer Situation aus, bei der die Tabelle Vortrigger, referenzielle Integritätsbedingungen, Prüfungen auf Integritätsbedingungen und Nachtrigger enthält, die hintereinander (kaskadierend) verarbeitet werden. Im Folgenden werden die Kästchen und anderen Elemente in Abb. 40 beschrieben.

- Anweisung S_1

  Dies ist die DELETE-, INSERT- oder UPDATE-Anweisung, die den Prozess erfordert. Die Anweisung S_1 gibt eine Tabelle (bzw. eine aktualisierbare Sicht für eine Tabelle) an, die in dieser Beschreibung als *Subjekttabelle* bezeichnet wird.
- Ermitteln der Menge der betroffenen Zeilen

  Dieser Schritt ist der Ausgangspunkt für einen Prozess, der für die Löschregeln CASCADE und SET NULL referenzieller Integritätsbedingungen sowie für kaskadierende Anweisungen aus Nachtriggern wiederholt ausgeführt wird.

  Zweck dieses Schritts ist die Ermittlung der *Menge der betroffenen Zeilen* (Set of Affected Rows - SAR) für die Anweisung. Die in dieser Menge enthaltenen Zeilen hängen von der Anweisung ab:
  - Für DELETE: Alle Zeilen, die die Suchbedingung der Anweisung erfüllen (oder die aktuelle Zeile bei einer positionierten DELETE-Operation)
  - Für INSERT: Die Zeilen, die durch die Klausel VALUES oder den Fullselect angegeben werden
  - Für UPDATE: Alle Zeilen, die die Suchbedingung erfüllten (oder die aktuelle Zeile bei einer positionierten UPDATE-Operation)

  Wenn die Menge der betroffenen Zeilen leer ist, sind keine Vortrigger, keine Änderungen zur Anwendung auf die Subjekttabelle und keine Integritätsbedingungen für die Anweisung zu verarbeiten.
- Verarbeiten der Vortrigger

  Alle Vortrigger (BEFORE) werden in der aufsteigenden Reihenfolge ihrer Erstellung verarbeitet. Jeder Vortrigger verarbeitet die ausgelöste Aktion einmal für jede Zeile in der Menge der betroffenen Zeilen.

Während der Verarbeitung einer ausgelösten Aktion kann ein Fehler auftreten. In diesem Fall werden alle Änderungen, die infolge der ursprünglichen Anweisung S_1 (bis dahin) ausgeführt wurden, rückgängig gemacht (Rollback).

Wenn keine Vortrigger definiert sind oder die Menge der betroffenen Zeilen leer ist, wird dieser Schritt übersprungen.

- Anwenden der Menge der betroffenen Zeilen auf die Subjekttabelle

  Die tatsächliche DELETE-, INSERT- oder UPDATE-Operation wird unter Verwendung der Menge der betroffenen Zeilen auf die Subjekttabelle (Zieltabelle) in der Datenbank angewendet.

  Beim Anwenden der Menge der betroffenen Zeilen kann ein Fehler auftreten (z. B. wenn versucht wird, eine Zeile mit einem bereits vorhandenen Schlüssel einzufügen, obwohl ein eindeutiger Index vorhanden ist). In diesem Fall werden alle Änderungen, die infolge der ursprünglichen Anweisung S_1 (bis dahin) ausgeführt wurden, rückgängig gemacht (Rollback).

- Anwenden der Integritätsbedingungen

  Die der Subjekttabelle zugeordneten Integritätsbedingungen werden angewendet, wenn die Menge der betroffenen Zeilen nicht leer ist. Dazu gehören eindeutige Integritätsbedingungen, eindeutige Indizes, referenzielle Integritätsbedingungen, Prüfungen auf Integritätsbedingungen und Prüfungen der Klausel WITH CHECK OPTION für Sichten. Referenzielle Integritätsbedingungen mit den Löschregeln CASCADE oder SET NULL können die Aktivierung weiterer Trigger bewirken.

  Ein Verstoß gegen eine Integritätsbedingung oder die Klausel WITH CHECK OPTION führt zu einem Fehler. In diesem Fall werden alle Änderungen, die infolge der ursprünglichen Anweisung S_1 (bis dahin) ausgeführt wurden, rückgängig gemacht (Rollback).

- Verarbeiten der Nachtrigger

  Alle von S_1 aktivierten Nachtrigger (AFTER) werden in der aufsteigenden Reihenfolge ihrer Erstellung verarbeitet.

  Trigger mit der Definition FOR EACH STATEMENT verarbeiten die ausgelöste Aktion genau einmal, auch wenn die Menge der betroffenen Zeilen leer ist. Trigger mit der Definition FOR EACH ROW verarbeiten die ausgelöste Aktion einmal für jede Zeile in der Menge der betroffenen Zeilen.

  Während der Verarbeitung einer ausgelösten Aktion kann ein Fehler auftreten. In diesem Fall werden alle Änderungen, die infolge der ursprünglichen Anweisung S_1 (bis dahin) ausgeführt wurden, rückgängig gemacht (Rollback).

  Die ausgelöste Aktion eines Triggers kann ausgelöste Anweisungen enthalten, bei denen es sich um DELETE-, INSERT- oder UPDATE-Anweisungen handelt. Zum Zweck dieser Beschreibung wird eine solche Anweisung als *kaskadierende Anweisung* betrachtet.

  Eine kaskadierende Anweisung ist eine DELETE-, INSERT- oder UPDATE-Anweisung, die als Teil der ausgelösten Aktion eines Nachtriggers verarbeitet wird. Eine solche Anweisung wird auf einer nachfolgenden Ebene der Triggerverarbeitung gestartet. Dies lässt sich in etwa so beschreiben, dass die ausgelöste Anweisung als neue Anweisung S_1 zugeordnet wird und alle hier beschriebenen Schritte wiederum rekursiv ausgeführt werden.

  Wenn alle ausgelösten Anweisungen aus allen Nachtriggern, die von jeder Anweisung S_1 aktiviert werden, vollständig verarbeitet wurden, wird die Verarbeitung der ursprünglichen Anweisung S_1 abgeschlossen.

- R = Rollback der Änderungen auf Stand vor S_1

  Jeder Fehler (einschließlich Verstößen gegen Integritätsbedingungen) während der Verarbeitung führt zu einem Rollback aller Änderungen, die direkt oder in-

direkt infolge der ursprünglichen Anweisung S_1 ausgeführt wurden. Die Datenbank wird in den Zustand zurückversetzt, den sie unmittelbar vor der Ausführung der ursprünglichen Anweisung S_1 hatte.

Fremdschlüssel in referenziellen Integritätsbedingungen

Ein Fremdschlüssel verweist auf einen Primärschlüssel oder einen eindeutigen Schlüssel in derselben oder einer anderen Tabelle. Die Zuordnung eines Fremdschlüssels zeigt an, dass die referenzielle Integrität gemäß der angegebenen referenziellen Integritätsbedingungen erhalten bleiben soll.

Ein Fremdschlüssel wird mit der Klausel FOREIGN KEY in der Anweisung CREATE TABLE bzw. ALTER TABLE definiert. Durch einen Fremdschlüssel wird die Tabelle von einer anderen Tabelle abhängig gemacht, die als übergeordnete Tabelle bezeichnet wird. Die Werte in der Spalte bzw. der Gruppe von Spalten, die den Fremdschlüssel in der einen Tabelle bilden, müssen mit den Werten des eindeutigen Schlüssels bzw. des Primärschlüssels der übergeordneten Tabelle übereinstimmen.

Die Anzahl der Spalten im Fremdschlüssel muss mit der Anzahl der Spalten im entsprechenden Primärschlüssel oder eindeutigen Schlüssel (auch als übergeordneter Schlüssel bezeichnet) der übergeordneten Tabelle übereinstimmen. Darüber hinaus müssen sich entsprechende Teile der Schlüsselspaltendefinitionen dieselben Datentypen und Datenlängen haben. Dem Fremdschlüssel kann ein *Integritätsbedingungsname* zugeordnet werden. Wenn Sie keinen Namen zuordnen, wird der Name automatisch zugeordnet. Zur leichteren Handhabung wird empfohlen, einen *Integritätsbedingungsnamen* zuzuordnen und nicht den vom System generierten Namen zu verwenden.

Der Wert eines zusammengesetzten Fremdschlüssels stimmt mit dem Wert eines Primärschlüssels überein, **wenn** der Wert jeder Spalte des Fremdschlüssels gleich dem Wert der entsprechenden Spalte des Primärschlüssels ist. Ein Fremdschlüssel, der Nullwerte enthält, kann nicht mit den Werten des Primärschlüssels übereinstimmen, da per Definition ein Primärschlüssels keine Nullwerte enthalten darf. Ein Nullwert eines Fremdschlüssels ist immer gültig, ungeachtet der Werte seiner Nichtnullwertspalten.

Für die Definition von Fremdschlüsseln gelten die folgenden Regeln:

- Eine Tabelle kann viele Fremdschlüssel enthalten.
- Ein Fremdschlüssel erfordert keine Eingabe (d. h. kann Nullwerte enthalten), wenn für irgendeinen seiner Teile keine Eingabe erforderlich ist (Nullwerte zulässig sind).
- Ein Fremdschlüsselwert ist ein Nullwert, wenn irgendeiner seiner Teile ein Nullwert ist.

Bei der Arbeit mit Fremdschlüsseln haben Sie folgende Möglichkeiten:

- Erstellen einer Tabelle ohne oder mit beliebig vielen Fremdschlüsseln
- Definieren von Fremdschlüsseln beim Erstellen oder Ändern einer Tabelle
- Löschen von Fremdschlüssels beim Ändern einer Tabelle

Auswirkungen von Tabellenintegritätsbedingungen auf Dienstprogrammoperationen

Wenn für die Tabelle, in die Daten geladen werden, referenzielle Integritätsbedingungen definiert sind, versetzt das Dienstprogramm LOAD die Tabelle in den Status 'Festlegen der Integrität anstehend', um Ihnen mitzuteilen, dass die Anweisung SET INTEGRITY für die Tabelle ausgeführt werden muss, um die referenzielle Inte-

grität der geladenen Zeilen zu prüfen. Nach Abschluss des Dienstprogramms LOAD müssen Sie die Anweisung SET INTEGRITY absetzen, um eine Prüfung der referenziellen Integrität für die geladenen Zeilen auszuführen und die Tabelle aus dem Status 'Festlegen der Integrität anstehend' herauszuholen.

Wenn beispielsweise die Tabellen DEPARTMENT und EMPLOYEE die einzigen Tabellen sind, die in den Status 'Festlegen der Integrität anstehend' gesetzt wurden, können Sie die folgende Anweisung ausführen:

```
SET INTEGRITY FOR DEPARTMENT ALLOW WRITE ACCESS,
   EMPLOYEE ALLOW WRITE ACCESS,
   IMMEDIATE CHECKED FOR EXCEPTION IN DEPARTMENT,
   USE DEPARTMENT_EX,
   IN EMPLOYEE USE EMPLOYEE_EX
```

Das Dienstprogramm zum Importieren (IMPORT) wird auf folgende Weise durch die referenziellen Integritätsbedingungen beeinflusst:

- Die Funktionen REPLACE und REPLACE CREATE sind nicht zulässig, wenn die Objekttabelle außer sich selbst noch andere abhängige Tabellen aufweist.

  Um diese Funktionen verwenden zu können, müssen erst alle Fremdschlüssel gelöscht werden, in denen die Tabelle eine übergeordnete Tabelle ist. Wenn die IMPORT-Operation abgeschlossen ist, erstellen Sie die Fremdschlüssel mit der Anweisung ALTER TABLE erneut.
- Der Erfolg eines Imports in eine Tabelle mit auf sich selbst verweisenden Integritätsbedingungen hängt von der Reihenfolge ab, in der die Zeilen importiert werden.

Anweisungsabhängigkeiten beim Ändern von Objekten

Anweisungsabhängigkeiten gelten für Pakete und im Cache zwischengespeicherte dynamische SQL- und XQuery-Anweisungen. Ein *Paket* ist ein Datenbankobjekt, das die Informationen enthält, die vom Datenbankmanager für den effizientesten Zugriff auf Daten für ein bestimmtes Anwendungsprogramm benötigt werden. Das *Binden* ist der Prozess, bei dem das Paket erstellt wird, das der Datenbankmanager für den Zugriff auf die Datenbank bei der Ausführung der Anwendung benötigt.

Pakete und im Cache zwischengespeicherte dynamische SQL- und XQuery-Anweisungen können von vielen Typen von Objekten abhängig sein.

Auf diese Objekte kann explizit verwiesen werden, zum Beispiel, indem eine Tabelle oder eine benutzerdefinierte Funktion in einer SQL-Anweisung SELECT angegeben wird. Auf die Objekte kann auch implizit verwiesen werden, zum Beispiel, wenn eine abhängige Tabelle überprüft werden muss, um sicherzustellen, dass keine **referenziellen Integritätsbedingungen** verletzt werden, wenn eine Zeile in einer übergeordneten Tabelle gelöscht wird. Pakete sind außerdem von den Zugriffsrechten abhängig, die dem Ersteller des Pakets erteilt sind.

Wenn ein Paket oder eine dynamische Abfrageanweisung von einem Objekt abhängig ist und dieses Objekt gelöscht wird, wird das Paket bzw. die im Cache zwischengespeicherte dynamische Abfrageanweisung in den Status „ungültig" (engl. invalid) versetzt. Wenn ein Paket von einer benutzerdefinierten Funktion abhängig ist und diese Funktion gelöscht wird, wird das Paket in den Status „funktionsunfähig" (engl. inoperative) mit folgenden Bedingungen versetzt:

- Eine im Cache zwischengespeicherte dynamische SQL- oder XQuery-Anweisung, die sich im Status „ungültig" befindet, wird bei ihrer nächsten Verwendung automatisch erneut optimiert. Wenn ein für die Anweisung erforderliches Objekt gelöscht wurde, schlägt die Ausführung der dynamischen SQL- oder XQuery-Anweisung möglicherweise mit einer entsprechenden Fehlernachricht fehl.

- Für ein Paket, das sich im Status „ungültig" befindet, wird bei der nächsten Verwendung ein impliziter Rebind durchgeführt. Für ein solches Paket kann aber auch ein expliziter Rebind durchgeführt werden. Wurde ein Paket als nicht gültig markiert, weil ein Trigger gelöscht wurde, ruft das erneut gebundene Paket keinen Trigger mehr auf.
- Für ein Paket, das sich im Status „funktionsunfähig" befindet, muss ein expliziter Rebind durchgeführt werden, damit es verwendet werden kann.

Für Objekte föderierter Datenbanken gelten ähnliche Abhängigkeiten. Durch das Löschen eines Servers oder das Ändern einer Serverdefinition werden zum Beispiel alle diesem Server zugeordneten Pakete oder im Cache zwischengespeicherte dynamische SQL-Anweisungen mit Verweisen auf Kurznamen ungültig gemacht (inaktiviert).

In einigen Fällen ist es nicht möglich, einen Rebind für das Paket durchzuführen. Zum Beispiel, wenn eine Tabelle gelöscht und nicht wieder erstellt wurde, kann für das Paket kein Rebind durchgeführt werden. In diesem Fall muss entweder das Objekt neu erstellt oder die Anwendung so geändert werden, dass sie nicht mehr auf das gelöschte Objekt zugreift.

In vielen anderen Fällen, zum Beispiel, wenn eine **Integritätsbedingung** gelöscht wurde, kann für das Paket ein Rebind durchgeführt werden.

Mithilfe der folgenden Systemkatalogsichten können Sie den Status eines Pakets und die Abhängigkeiten des Pakets feststellen:

- SYSCAT.PACKAGEAUTH
- SYSCAT.PACKAGEDEP
- SYSCAT.PACKAGES

Entwerfen von informativen Integritätsbedingungen

Integritätsbedingungen, die der Datenbankmanager beim Einfügen oder Aktualisieren von Datensätzen umsetzt, können zu einer hohen Systemaktivität führen, besonders dann, wenn große Mengen an Datensätzen geladen werden, die über referenzielle Integritätsbedingungen verfügen. Wenn eine Anwendung Informationen bereits vor dem Einfügen eines Datensatzes in die Tabelle überprüft hat, ist die Verwendung von informativen Integritätsbedingungen möglicherweise effizienter als die Verwendung normaler Integritätsbedingungen.

Informative Integritätsbedingungen teilen dem Datenbankmanager mit, welchen Regeln die Daten entsprechen; allerdings werden die Regeln nicht vom Datenbankmanager umgesetzt. Diese Informationen können jedoch vom DB2-Optimierungsprogramm verwendet werden und zu einer besseren Leistung von SQL-Abfragen beitragen.

Im folgenden Beispiel wird die Verwendung von informativen Integritätsbedingungen und deren Funktionsweise dargestellt. In dieser einfachen Tabelle sind Informationen zum Alter und zum Geschlecht des Antragstellers enthalten:

```
CREATE TABLE APPLICANTS
  (
   AP_NO INT NOT NULL,
     GENDER  CHAR(1) NOT NULL,
CONSTRAINT GENDEROK
   CHECK (GENDER IN ('M', 'F'))
   NOT ENFORCED
   ENABLE QUERY OPTIMIZATION,
      AGE  INT NOT NULL,
```

```
CONSTRAINT AGEOK
   CHECK (AGE BETWEEN 1 AND 80)
   NOT ENFORCED
   ENABLE QUERY OPTIMIZATION,
 );
```

In diesem Beispiel sind zwei Klauseln enthalten, die das Verhalten der Integritätsbedingungen für Spalten ändern. Bei der ersten Option handelt es sich um die Option NOT ENFORCED; diese weist den Datenbankmanager an, die Prüfung dieser Spalte nicht umzusetzen, wenn Daten eingefügt oder aktualisiert werden. Für diese Option kann zusätzlich TRUSTED oder NOT TRUSTED angegeben werden. Wenn für die informative Integritätsbedingung die Einstellung TRUSTED angegeben ist, kann der Datenbankmanager davon ausgehen, dass die Daten der Integritätsbedingung entsprechen. Dies ist die Standardoption. Wenn NOT TRUSTED angegeben ist, geht der Datenbankmanager davon aus, dass die meisten der Daten (jedoch nicht alle) nicht der Integritätsbedingung entsprechen. In diesem Beispiel wird standardmäßig die Option NOT ENFORCED TRUSTED verwendet, da keine der beiden Optionen TRUSTED oder NOT TRUSTED angegeben wurde.

Bei der zweiten Option handelt es sich um die Option ENABLE QUERY OPTIMIZATION, die vom Datenbankmanager verwendet wird, wenn SELECT-Anweisungen für diese Tabelle ausgeführt werden. Wird dieser Wert angegeben, verwendet der Datenbankmanager bei der SQL-Optimierung die Informationen in der Integritätsbedingung.

Wenn in der Tabelle die Option NOT ENFORCED enthalten ist, kann das Verhalten von Einfügeanweisungen ungewohnt erscheinen. Die folgende SQL-Anweisung schlägt nicht fehl, wenn sie für die Tabelle APPLICANTS ausgeführt wird:

```
INSERT INTO APPLICANTS VALUES
 (1, 'M', 54),
 (2, 'F', 38),
 (3, 'M', 21),
 (4, 'F', 89),
 (5, 'C', 10),
 (6, 'S',100),
```

Der Antragsteller Nummer fünf hat den Wert C für GENDER (C = Child), und der Antragsteller Nummer sechs hat einen ungültigen Wert für GENDER (S = Senior) und überschreitet die Altersbegrenzung der Spalte AGE. In beiden Fällen ermöglicht der Datenbankmanager die Einfügeoperation, da für die Integritätsbedingungen NOT ENFORCED und TRUSTED angegeben ist. Im folgenden Beispiel sehen Sie das Ergebnis einer SELECT-Anweisung für die Tabelle:

```
SELECT * FROM APPLICANTS
  WHERE GENDER = 'C';

APPLICANT  GENDER  AGE
---------  ------  ---

0 Satz/Sätze ausgewählt.
```

Der Datenbankmanager hat die falsche Antwort auf die Abfrage zurückgegeben, obwohl der Wert 'C' in der Tabelle gefunden wurde; allerdings wird dem Datenbankmanager über die Integritätsbedingung dieser Spalte mitgeteilt, dass der einzig gültige Wert 'M' bzw. 'F' ist. Auch das Schlüsselwort ENABLE QUERY OPTIMIZATION ermöglichte dem Datenbankmanager bei der Optimierung der Anweisung die Verwendung dieser Integritätsbedingungsinformationen. Wenn dies nicht das erwünschte Verhalten ist, muss die Integritätsbedingung mithilfe der Anweisung ALTER TABLE wie im folgenden Beispiel geändert werden:

```
ALTER TABLE APPLICANTS
  ALTER CHECK AGEOK DISABLE QUERY OPTIMIZATION
```

Wenn die Abfrage erneut abgesetzt wird, gibt der Datenbankmanager die folgenden richtigen Ergebnisse zurück:

```
SELECT * FROM APPLICANTS
  WHERE SEC = 'C';

APPLICANT  GENDER  AGE
---------  ------  ---
        5  C       10

1 Satz/Sätze ausgewählt.
```

Anmerkung: Wären die Integritätsbedingungsattribute NOT ENFORCED NOT TRUSTED und ENABLE QUERY OPTIMIZATION von Anfang an für die Tabelle APPLICANTS angegeben wurden, wären die zuvor angezeigten korrekten Ergebnisse nach dem Absetzen der ersten SELECT-Anweisung zurückgegeben worden.

Das beste Szenario für die Verwendung von informativen Integritätsbedingungen mit dem Attribut NOT ENFORCED TRUSTED ist dann gegeben, wenn Sie garantierten können, dass es sich bei dem Anwendungsprogramm um die einzige Anwendung handelt, die die Daten einfügt und aktualisiert. Wenn die Anwendung bereits alle Informationen im Vorfeld prüft (z. B. Geschlecht und Alter im obigen Beispiel), kann sich aus der Verwendung von informativen Integritätsbedingungen eine schnellere Leistung und ein um die Hälfte reduzierter Aufwand ergeben. Ein weiterer möglicher Einsatzbereich informativer Integritätsbedingungen ist das Entwerfen von Data-Warehouses. Auch wenn Sie nicht garantieren können, dass die Daten in der Tabelle stets der Integritätsbedingung entsprechen, können Sie für die Integritätsbedingungen die Attribute NOT ENFORCED und NOT TRUSTED angeben. Dieser Typ von Integritätsbedingung kann verwendet werden, wenn die strikte Übereinstimmung zwischen den Werten der Fremdschlüssel und der Primärschlüssel nicht erforderlich ist. Außerdem kann diese Integritätsbedingung auch weiterhin als Teil einer Statistiksicht verwendet werden, um die Optimierung bestimmter SQL-Abfragen zu ermöglichen.

Erstellen und Ändern von Integritätsbedingungen

Integritätsbedingungen können vorhandenen Tabellen mithilfe der Anweisung ALTER TABLE hinzugefügt werden.

Informationen zu diesem Vorgang

Der Name der Integritätsbedingung darf nicht mit dem Namen einer anderen Integritätsbedingung innerhalb einer Anweisung ALTER TABLE übereinstimmen und muss innerhalb der Tabelle eindeutig sein (dies schließt auch Namen bereits definierter referenzieller Integritätsbedingungen mit ein). Vorhandene Daten werden anhand der neuen Bedingung überprüft, bevor die Anweisung erfolgreich ausgeführt wird.

Erstellen und Ändern eindeutiger Integritätsbedingungen

Eindeutige Integritätsbedingungen können einer vorhandenen Tabelle hinzugefügt werden. Der Name der Integritätsbedingung darf nicht mit dem Namen einer anderen Integritätsbedingung innerhalb derselben Anweisung ALTER TABLE übereinstimmen und muss innerhalb der Tabelle eindeutig sein (dies schließt auch Namen bereits definierter referenzieller Integritätsbedingungen mit ein). Vorhandene Daten werden anhand der neuen Bedingung überprüft, bevor die Anweisung erfolgreich ausgeführt wird.

Zur Definition von eindeutigen Integritätsbedingungen über die Befehlszeile verwenden Sie die Option ADD CONSTRAINT der Anweisung ALTER TABLE. Mit der folgenden Anweisung wird der Tabelle EMPLOYEE zum Beispiel eine eindeutige Integritätsbedingung hinzugefügt, die eine neue Methode zur eindeutigen Identifizierung von Mitarbeitern in der Tabelle darstellt:

```
ALTER TABLE EMPLOYEE
   ADD CONSTRAINT NEWID UNIQUE(EMPNO,HIREDATE)
```

Wenn Sie diese Integritätsbedingung ändern wollen, müssen Sie sie löschen und anschließend erneut erstellen.

Erstellen und Ändern von Integritätsbedingungen über Primärschlüssel

Einer vorhandenen Tabelle kann eine Integritätsbedingung über Primärschlüssel hinzugefügt werden. Der Name der Integritätsbedingung muss innerhalb der Tabelle (einschließlich der Namen aller definierten referenziellen Integritätsbedingungen) eindeutig sein. Vorhandene Daten werden anhand der neuen Bedingung überprüft, bevor die Anweisung erfolgreich ausgeführt wird.

Geben Sie die folgende Anweisung in die Befehlszeile ein, um Primärschlüssel hinzuzufügen:

```
ALTER TABLE <name>
   ADD CONSTRAINT <spaltenname>
   PRIMARY KEY <spaltenname>
```

Eine vorhandene Integritätsbedingung kann nicht geändert werden. Wenn Sie eine andere Spalte bzw. eine andere Gruppe von Spalten als Primärschlüssel definieren wollen, muss die vorhandene Primärschlüsseldefinition zunächst gelöscht und anschließend erneut erstellt werden.

Erstellen und Ändern von Prüfungen auf Integritätsbedingungen

Wenn eine Prüfung auf Integritätsbedingung in einer Tabelle hinzugefügt wird, werden Pakete und im Cache zwischengespeicherte dynamische SQL-Anweisungen, die INSERT- oder UPDATE-Operationen für die Tabelle ausführen, eventuell als ungültig markiert.

Geben Sie die folgende Anweisung in die Befehlszeile ein, um eine Prüfung auf Integritätsbedingung in der Tabelle hinzuzufügen:

```
ALTER TABLE EMPLOYEE
   ADD CONSTRAINT REVENUE CHECK (SALARY + COMM > 25000)
```

Wenn Sie diese Integritätsbedingung ändern wollen, müssen Sie sie löschen und anschließend erneut erstellen.

Erstellen und Ändern von (referenziellen) Integritätsbedingungen über Fremdschlüssel

Ein Fremdschlüssel ist ein Verweis auf die Datenwerte in einer anderen Tabelle. Es gibt verschiedene Typen von Integritätsbedingungen über Fremdschlüssel.

Wenn ein Fremdschlüssel einer Tabelle hinzugefügt wird, werden Pakete und im Cache zwischengespeicherte dynamische SQL-Anweisungen, die folgende Arten von Anweisungen enthalten, eventuell als ungültig markiert:

- Anweisungen, die die Tabelle mit dem Fremdschlüssel aktualisieren oder dort Zeilen einfügen
- Anweisungen, die die übergeordnete Tabelle aktualisieren oder löschen

Geben Sie die folgende Anweisung in die Befehlszeile ein, um Fremdschlüssel hinzuzufügen:

```
ALTER TABLE <name>
   ADD CONSTRAINT <spaltenname>
    FOREIGN KEY <spaltenname>
    ON DELETE <aktionstyp>
    ON UPDATE <aktionstyp>
```

Die folgenden Beispiele zeigen die Verwendung der Anweisung ALTER TABLE zum Hinzufügen von Primär- und Fremdschlüsseln in einer Tabelle:

```
ALTER TABLE PROJECT
  ADD CONSTRAINT PROJECT_KEY
      PRIMARY KEY (PROJNO)
ALTER TABLE EMP_ACT
  ADD CONSTRAINT ACTIVITY_KEY
      PRIMARY KEY (EMPNO, PROJNO, ACTNO)
  ADD CONSTRAINT ACT_EMP_REF
      FOREIGN KEY (EMPNO)
      REFERENCES EMPLOYEE
      ON DELETE  RESTRICT
  ADD CONSTRAINT ACT_PROJ_REF
      FOREIGN KEY (PROJNO)
      REFERENCES PROJECT
      ON DELETE CASCADE
```

Wenn Sie diese Integritätsbedingung ändern wollen, müssen Sie sie löschen und anschließend erneut erstellen.

Erstellen und Ändern informativer Integrationsbedingungen

Zur Verbesserung der Leistung von Abfragen können Sie Ihren Tabellen informative Integritätsbedingungen hinzufügen. Sie können informative Integrationsbedingungen mithilfe der Anweisung CREATE TABLE oder ALTER TABLE hinzufügen, wenn Sie die Option `NOT ENFORCED` in der DDL-Anweisung angeben. Mit der Option NOT ENFORCED können Sie außerdem TRUSTED oder NOT TRUSTED für die Integritätsbedingung angeben.

Einschränkung: Nach der Definition informativer Integritätsbedingungen in einer Tabelle können die Spaltennamen für diese Tabelle erst geändert werden, wenn die informativen Integritätsbedingungen entfernt wurden.

Zur Angabe informativer Integrationsbedingungen für eine Tabelle über die Befehlszeile geben Sie einen der folgenden Befehle für eine neue Tabelle ein:

```
ALTER TABLE <name> <attribute_der_integritätsbedingung> NOT ENFORCED
ALTER TABLE <name> <attribute_der_integritätsbedingung> NOT ENFORCED TRUSTED
ALTER TABLE <name> <attribute_der_integritätsbedingung> NOT ENFORCED NOT TRUSTED
```

ENFORCED oder NOT ENFORCED: Gibt an, ob die Integritätsbedingung vom Datenbankmanager während normaler Operationen, wie z. B. Einfüge-, Aktualisierungs- oder Löschoperationen, erzwungen wird.

- ENFORCED kann nicht für eine funktionale Abhängigkeit angegeben werden (SQLSTATE-Wert 42621).
- NOT ENFORCED sollte nur angegeben werden, wenn die Tabellendaten der Integritätsbedingung entsprechen, ohne dass dies erzwungen wurde. Abfrageergebnisse können unvorhersehbar sein, wenn die Daten nicht tatsächlich der Integritätsbedingung entsprechen. Ferner können Sie für die Integritätsbedingung NOT ENFORCED entweder TRUSTED oder NOT TRUSTED angeben.

– TRUSTED: Gibt an, dass der Datenbankmanager darauf vertrauen kann, dass die Daten der Integritätsbedingung entsprechen. Dies ist die Standardoption. Diese Option darf nur verwendet werden, wenn die Daten der Integritätsbedingung entsprechen, ohne dass dies erzwungen wurde.
– NOT TRUSTED: Gibt an, dass der Datenbankmanager nicht darauf vertrauen kann, dass die Daten der Integritätsbedingung entsprechen. Diese Option wird für Fälle verwendet, in denen die Daten der Integritätsbedingung für die Mehrzahl der Zeilen entsprechen, die Entsprechung aber erzwungen wurde. NOT TRUSTED kann nur für referenzielle Integritätsbedingungen (SQLSTATE 42613) angegeben werden.

Wenn Sie diese Integritätsbedingung ändern wollen, müssen Sie sie löschen und anschließend erneut erstellen.

Wiederverwendung von Indizes bei Integritätsbedingungen über eindeutigen Schlüssel und Primärschlüssel

Wenn Sie einer partitionierten Tabelle mit einem partitionierten Index mithilfe der Anweisung ALTER TABLE eine Integritätsbedingung über eindeutigen Schlüssel oder eine Integritätsbedingung über Primärschlüssel hinzufügen, wird je nach den bereits vorhandenen Indizes einer von diesen zur Umsetzung der neuen Integritätsbedingung geändert oder ein neuer Index erstellt.

Wenn Sie die Anweisung ALTER TABLE ausführen, um einen eindeutigen Schlüssel oder einen Primärschlüssel für eine Tabelle hinzuzufügen oder zu ändern, wird eine Prüfung ausgeführt, um festzustellen, ob ein vorhandener Index dem eindeutigen Schlüssel bzw. dem Primärschlüssel entspricht, der definiert wird. (INCLUDE-Spalten werden nicht berücksichtigt.) Eine Indexdefinition entspricht einem solchen Schlüssel, wenn sie dieselbe Gruppe von Spalten angibt, wobei die Reihenfolge oder die Suchrichtung der Spalten (z. B. ASC/DESC - Auf-/Absteigend) keine Rolle spielt.

Bei partitionierten Tabellen, die partitionierte, nicht eindeutige Indizes haben, wird ein Index nicht als ein den Schlüsseln entsprechender Index betrachtet, wenn sich die Indexspalten der Tabelle, die geändert wird, nicht unter den Spalten befinden, die den Partitionsschlüssel bilden.

Wenn die Tabelle eine entsprechende Indexdefinition hat, wird der Index in einen eindeutigen (UNIQUE) Index geändert, wenn er nicht bereits ein eindeutiger Index war, und wird als für das System erforderlich markiert. Wenn für die Tabelle mehrere Indizes vorhanden sind, die den Schlüsseln entsprechen, wird ein vorhandener eindeutiger Index ausgewählt. Wenn mehrere eindeutige Indizes vorhanden sind, die dem Schlüssel entsprechen, oder wenn mehrere nicht eindeutige Indizes, jedoch kein dem Schlüssel entsprechender eindeutiger Index vorhanden sind, wird ein partitionierter Index bevorzugt. Anderenfalls erfolgt die Auswahl eines Index beliebig.

Wenn kein entsprechender Index gefunden wird, wird automatisch ein eindeutiger, bidirektionaler Index für die Spalten erstellt.

Anzeigen der Definitionen von Integritätsbedingungen für eine Tabelle

Definitionen von Integritätsbedingungen für eine Tabelle befinden sich in den Katalogsichten SYSCAT.INDEXES und SYSCAT.REFERENCES.

Informationen zu diesem Vorgang

Die Spalte UNIQUERULE der Sicht SYSCAT.INDEXES gibt das Merkmal des Index an. Wenn der Wert dieser Spalte 'P' ist, handelt es sich bei dem Index um einen Primärschlüsselindex. Ist der Wert 'U', handelt es sich um einen eindeutigen Index (jedoch nicht für einen Primärschlüssel).

Die Katalogsicht SYSCAT.REFERENCES enthält Informationen zu referenziellen Integritätsbedingungen (über Fremdschlüssel).

Löschen von Integritätsbedingungen

Eine Prüfung auf Integritätsbedingung in einer Tabelle (Table Check Constraint) kann explizit mithilfe einer Anweisung ALTER TABLE oder implizit als Ergebnis einer Anweisung DROP TABLE gelöscht werden.

Informationen zu diesem Vorgang

Zum Löschen von Integritätsbedingungen verwenden Sie die Anweisung ALTER TABLE mit der Klausel DROP oder DROP CONSTRAINT. Dadurch können Sie die Tabellen, die die betreffenden Spalten enthalten, binden (BIND) und weiterhin auf sie zugreifen. Die Namen aller eindeutigen Integritätsbedingungen für eine Tabelle befinden sich in der Systemkatalogsicht SYSCAT.INDEXES.

Vorgehensweise

- Mit der Klausel DROP UNIQUE der Anweisung ALTER TABLE können Sie eindeutige Integritätsbedingungen explizit löschen.

  Die Klausel DROP UNIQUE der Anweisung ALTER TABLE löscht die Definition der eindeutigen Integritätsbedingung *name_der_eindeutigen_integritätsbedingung* sowie alle referenziellen Integritätsbedingungen, die von dieser eindeutigen Integritätsbedingung abhängig sind. Mit *name_der_eindeutigen_integritätsbedingung* muss eine vorhandene eindeutige Integritätsbedingung angegeben werden.

  ```
      ALTER TABLE tabellenname
  DROP UNIQUE name_der_eindeutigen_integritätsbedingung
  ```

  Durch das Löschen dieser eindeutigen Integritätsbedingung werden alle Pakete und im Cache zwischengespeicherte dynamische SQL-Anweisungen ungültig gemacht, die diese Integritätsbedingung verwendet haben.
- Mit der Klausel DROP PRIMARY KEY der Anweisung ALTER TABLE können Sie Integritätsbedingungen über Primärschlüssel löschen.

  Die Klausel DROP PRIMARY KEY der Anweisung ALTER TABLE löscht die Definition des Primärschlüssels sowie alle referenziellen Integritätsbedingungen, die von diesem Primärschlüssel abhängig sind. Die Tabelle muss einen Primärschlüssel besitzen. Geben Sie die folgende Anweisung in die Befehlszeile ein, um einen Primärschlüssel zu löschen:

  ```
      ALTER TABLE tabellenname
  DROP PRIMARY KEY
  ```
- Mit der Klausel DROP CHECK der Anweisung ALTER TABLE können Sie Prüfungen auf Integritätsbedingung (für Tabellen) löschen.

Durch das Löschen einer Prüfung auf Integritätsbedingung werden alle Pakete und im Cache zwischengespeicherte dynamische Anweisungen mit INSERT- oder UPDATE-Abhängigkeiten für die Tabelle ungültig gemacht. Die Namen aller Prüfungen auf Integritätsbedingungen in einer Tabelle können mithilfe der Katalogsicht SYSCAT.CHECKS festgestellt werden. Vor dem Löschen einer Prüfung auf Integritätsbedingungen in einer Tabelle, die einen vom System generierten Namen hat, können Sie den Namen aus der Katalogsicht SYSCAT.CHECKS abfragen.

Mit der folgenden Anweisung wird die Prüfung auf Integritätsbedingung *name_der_prüfung_auf_integritätsbedingung* gelöscht. Der Wert für *name_der_prüfung_auf_integritätsbedingung* muss eine vorhandene Prüfung auf Integritätsbedingung bezeichnen, die für die Tabelle definiert ist. Geben Sie die folgende Anweisung in die Befehlszeile ein, um eine Prüfung auf Integritätsbedingung in der Tabelle zu löschen:

```
    ALTER TABLE tabellenname
DROP CHECK name_der_prüfung_auf_integritätsbedingung
```

Alternativ dazu können Sie die Anweisung ALTER TABLE mit der Option DROP CONSTRAINT verwenden.

- Mit der Klausel DROP CONSTRAINT der Anweisung ALTER TABLE können Sie (referenzielle) Integritätsbedingungen über Fremdschlüssel löschen.

  Die Klausel DROP CONSTRAINT der Anweisung ALTER TABLE löscht die Integritätsbedingung mit dem angegeben Namen (*integritätsbedingungsname*). Der angegebene Name *integritätsbedingungsname* muss eine vorhandene Integritätsbedingung über Fremdschlüssel, eine Integritätsbedingung über Primärschlüssel oder eine eindeutige Integritätsbedingung bezeichnen, die für die Tabelle definiert ist. Geben Sie die folgende Anweisung in die Befehlszeile ein, um Fremdschlüssel zu löschen:

  ```
      ALTER TABLE tabellenname
  DROP FOREIGN KEY fremdschlüsselname
  ```

  Wenn eine Integritätsbedingung über Fremdschlüssel gelöscht wird, werden Pakete oder im Cache zwischengespeicherte dynamische Anweisungen, die folgende Arten von Anweisungen enthalten, eventuell als ungültig markiert:

  – Anweisungen, die die Tabelle mit dem Fremdschlüssel aktualisieren oder dort Zeilen einfügen
  – Anweisungen, die die übergeordnete Tabelle aktualisieren oder löschen

Beispiel

Im folgenden Beispiel werden die Klauseln DROP PRIMARY KEY und DROP FOREIGN KEY in der Anweisung ALTER TABLE verwendet, um Primär- und Fremdschlüssel zu löschen:

```
ALTER TABLE EMP_ACT
  DROP PRIMARY KEY
  DROP FOREIGN KEY ACT_EMP_REF
  DROP FOREIGN KEY ACT_PROJ_REF
ALTER TABLE PROJECT
  DROP PRIMARY KEY
```

Kapitel 14. Indizes

Ein *Index* ist eine Gruppe von Verweisen, die logisch nach den Werten eines oder mehrerer Schlüssel sortiert sind. Die Verweise können sich auf Zeilen in einer Tabelle, Blöcke in einer MDC- oder ITC-Tabelle, XML-Daten in einem XML-Speicherobjekt usw. beziehen.

Indizes dienen folgenden Zwecken:

- Verbesserung der Leistung. In den meisten Fällen kann mithilfe eines Index schneller auf Daten zugegriffen werden. Zwar kann ein Index nicht für eine Sicht erstellt werden, aber ein für die Tabelle, auf der eine Sicht basiert, erstellter Index kann gelegentlich die Leistung von Operationen für diese Sicht verbessern.
- Sicherstellung der Eindeutigkeit. Eine Tabelle mit einem eindeutigen Index kann nicht mehrere Zeilen mit identischen Schlüsseln haben.

Wenn Daten einer Tabelle hinzugefügt werden, werden sie am Ende der Tabelle angefügt (sofern keine anderen Aktionen an der Tabelle bzw. an den eingefügten Daten ausgeführt wurden). Für die Daten gibt es keine inhärente Reihenfolge. Wenn nach einer bestimmten Datenzeile gesucht wird, muss von der ersten bis zur letzten Zeile jede Zeile der Tabelle überprüft werden. Indizes werden als Methode zum Zugreifen auf die Daten in der Tabelle in einer Reihenfolge verwendet, die möglicherweise andernfalls nicht verfügbar ist.

Wenn Sie in einer Tabelle nach Daten suchen, suchen Sie in der Regel nach Zeilen mit Spalten, die bestimmte Werte enthalten. Ein Spaltenwert in einer Datenzeile kann zum Kennzeichnen der gesamten Zeile verwendet werden. Es ist zum Beispiel wahrscheinlich, dass eine Personalnummer einen bestimmten Mitarbeiter eindeutig definiert. Andererseits könnte auch mehr als eine Spalte zur Identifikation der Zeile erforderlich sein. Dies könnte zum Beispiel eine Kombination aus Kundenname und Telefonnummer sein. Spalten in einem Index, die zur Identifikation von Datenzeilen verwendet werden, werden als *Schlüssel* bezeichnet. Eine Spalte kann in mehr als einem Schlüssel verwendet werden.

Ein Index wird nach den Werten in einem Schlüssel sortiert. Schlüssel können eindeutig oder nicht eindeutig sein. Jede Tabelle muss über mindestens einen eindeutigen Schlüssel verfügen; allerdings können die Tabellen aber auch über weitere nicht eindeutige Schlüssel verfügen. Jeder Index verfügt über genau einen Schlüssel. Beispiel: Sie können die Mitarbeiter-ID-Nummer (eindeutig) als Schlüssel für einen Index und die Abteilungsnummer (nicht eindeutig) als Schlüssel für einen anderen Index verwenden.

Nicht alle Indizes verweisen auf Zeilen in einer Tabelle. MDC- und ITC-Blockindizes verweisen auf Speicherbereiche (bzw. Blöcke) von Daten. XML-Indizes für XML-Daten verwenden bestimmte XML-Musterausdrücke zur Indexierung von Pfaden und Werten in XML-Dokumenten, die innerhalb einer Spalte gespeichert sind. Der Datentyp einer solchen Spalte muss XML sein. Sowohl MDC- und ITC-Blockindizes als auch XML-Indizes sind systemgenerierte Indizes.

Beispiel

Tabelle A in Abb. 41 verfügt über einen Index, der auf den Personalnummern in der Tabelle basiert. Dieser Schlüsselwert dient als Zeiger auf die Zeilen in der Tabelle. Beispiel: Die Personalnummer 19 zeigt auf den Mitarbeiter KMP. Ein Index ermöglicht einen effizienten Zugriff auf Zeilen einer Tabelle, indem er einen Pfad zu den Daten mithilfe von Zeigern erstellt.

Es können eindeutige Indizes erstellt werden, um die Eindeutigkeit des Indexschlüssels sicherzustellen. Ein *Indexschlüssel* ist eine Spalte oder eine geordnete Gruppe von Spalten, auf denen ein Index definiert wird. Mithilfe eines eindeutigen Index wird gewährleistet, dass der Wert jedes Indexschlüssels in der indizierten Spalte bzw. den indizierten Spalten eindeutig ist.

Abb. 41 illustriert die Beziehung zwischen einem Index und einer Tabelle.

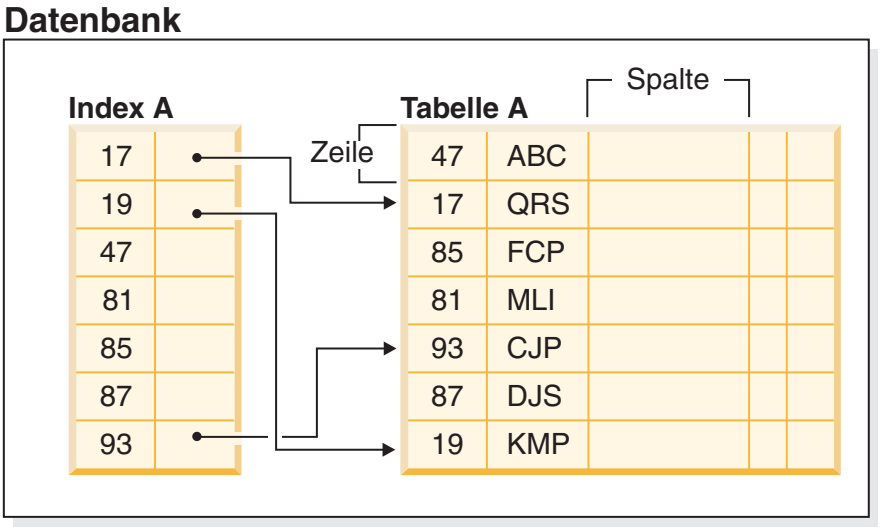


Abbildung 41. Beziehung zwischen einem Index und einer Tabelle

Abb. 42 auf Seite 477 veranschaulicht die Beziehungen zwischen einigen Datenbankobjekten. Sie zeigt außerdem, dass Tabellen, Indizes und lange Daten (LOB-Daten) in Tabellenbereichen gespeichert werden.

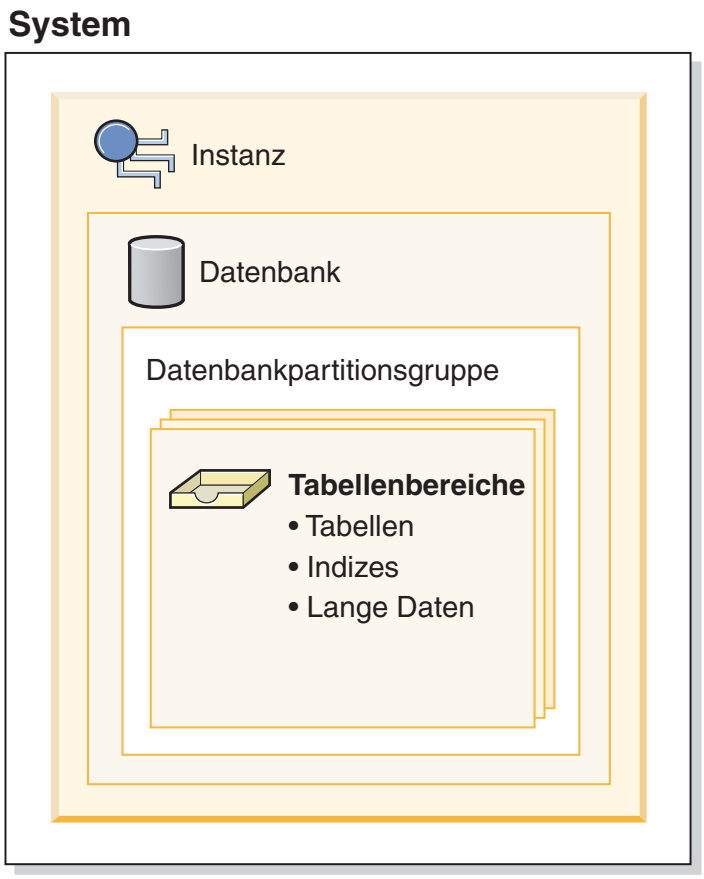


Abbildung 42. Beziehung zwischen ausgewählten Datenbankobjekten

Typen von Indizes

Es gibt verschiedene Typen von Indizes, die zu verschiedenen Zwecken erstellt werden können. Eindeutige Indizes dienen zum Beispiel zur Umsetzung der Integritätsbedingung der Eindeutigkeit in den Indexschlüsseln. Bidirektionale Indizes ermöglichen ein Durchsuchen des Index in beiden Richtungen (vorwärts und rückwärts). Clusterindizes helfen, die Leistung von Abfragen zu verbessern, bei denen die entsprechende Tabelle in der Reihenfolge der Schlüssel durchsucht wird.

Eindeutige und nicht eindeutige Indizes

Bei eindeutigen Indizes handelt es sich um Indizes, die zur Pflege der Datenintegrität beitragen, indem sie sicherstellen, dass keine zwei Datenzeilen in einer Tabelle über identische Schlüsselwerte verfügen.

Bei dem Versuch, einen eindeutigen Index für eine Tabelle zu erstellen, die bereits Daten enthält, werden Werte in der/den Spalten(n), aus der/denen der Index besteht, auf ihre Eindeutigkeit hin überprüft. Wenn die Tabelle Zeilen mit doppelten Schlüsselwerten enthält, schlägt die Indexerstellung fehl. Wenn ein eindeutiger Index für eine Tabelle definiert wurde, wird die Eindeutigkeit sichergestellt, wenn Schlüssel im Index hinzugefügt oder geändert werden. (Dies gilt zum Beispiel für Operationen wie INSERT, UPDATE, LOAD, IMPORT und SET INTEGRITY u. a.)

Zusätzlich zur Gewährleistung der Eindeutigkeit von Datenwerten kann ein eindeutiger Index auch zur Verbesserung der Leistung beim Datenabruf während der Abfrageverarbeitung verwendet werden.

Nicht eindeutige Indizes werden nicht für die Umsetzung von Integritätsbedingungen für die Tabellen verwendet, denen sie zugeordnet sind. Stattdessen werden nicht eindeutige Indizes ausschließlich zur Verbesserung der Abfrageleistung durch Beibehalten einer sortierten Reihenfolge von Datenwerten verwendet, die häufig benutzt werden.

Cluster- und Nicht-Clusterindizes

Indexarchitekturen werden als Cluster- und Nicht-Clusterindexarchitekturen klassifiziert. Clusterindizes sind Indizes, bei denen die Reihenfolge der Zeilen in den Datenseiten der Reihenfolge der Zeilen im Index entspricht. Dies der Grund, aus dem es nur einen Clusterindex für eine bestimmte Tabelle geben kann, während die Tabelle viele Nicht-Clusterindizes haben kann. Bei einigen Verwaltungssystemen für relationale Datenbanken entspricht der Blattknoten (Leaf Node) des Clusterindex den eigentlichen Daten und nicht einem Zeiger auf Daten, die sich an anderer Stelle befinden.

Clusterindizes und Nicht-Clusterindizes können nur Schlüssel und Satz-IDs (RIDs) in der Indexstruktur enthalten. Die Satz-IDs zeigen immer auf Zeilen in den Datenseiten. Der einzige Unterschied zwischen einem Clusterindex und einem Nicht-Clusterindex besteht darin, dass der Datenbankmanager versucht, die Daten in den Datenseiten in genau der Reihenfolge zu halten, in der die entsprechenden Schlüssel in den Indexseiten angeordnet sind. Daher versucht der Datenbankmanager, Zeilen mit ähnlichen Schlüsseln in dieselben Seiten einzufügen. Wenn die Tabelle reorganisiert wird, werden ihre Zeilen in der Reihenfolge der Inderschlüssel in die Datenseiten eingefügt.

Die Reorganisation einer Tabelle in Bezug auf einen ausgewählten Index dient zur erneuten Erstellung des Datenclusterings. Ein Clusterindex eignet sich am Besten für Spalten, die über Bereichsvergleichselemente verfügen, da dieser einen verbesserten sequenziellen Zugriff auf die Daten der Tabelle ermöglicht. Dadurch werden weniger Seitenabrufoperationen benötigt, weil sich ähnliche Werte in derselben Datenseite befinden.

Im Allgemeinen kann nur einer der Indizes in einer Tabelle einen hohen Grad an Clusterbildung aufweisen.

Mithilfe von Clusterindizes kann die Leistung der meisten Abfrageoperationen verbessert werden, da sie einen linearen Zugriffspfad zu den Daten bereitstellen, die in Seiten gespeichert wurden. Außerdem ist der Vorablesezugriff mit Sequenzerkennung in der Regel effizienter, wenn Clusterindizes verwendet werden, da Zeilen mit ähnlichen Indexschlüsselwerten zusammen gespeichert werden.

Allerdings können Clusterindizes nicht im Rahmen der Tabellendefinition angegeben werden, die mit der Anweisung CREATE TABLE verwendet wird. Stattdessen werden Clusterindizes nur durch die Ausführung der Anweisung CREATE INDEX mit der Option CLUSTER erstellt. Dann muss die Anweisung ALTER TABLE zum Hinzufügen eines Primärschlüssels verwendet werden, der dem für die Tabelle erstellten Clusterindex entspricht. Dieser Clusterindex wird dann als Primärschlüsselindex der Tabelle verwendet.

Anmerkung: Das Einstellen von PCTFREE in der Tabelle mit der Anweisung ALTER TABLE auf einen geeigneten Wert kann helfen, das Clustering der Tabelle aufrechtzuerhalten, indem ausreichend freier Speicherbereich zum Einfügen von Zeilen in den Seiten mit ähnlichen Werten freigehalten wird. Weitere Informationen finden Sie in der Beschreibung der „Anweisung ALTER TABLE" im Handbuch *SQL Reference* sowie im Abschnitt „Senken des Bedarfs an Tabellen- und Indexreorganisationen" im Handbuch *Fehlerbehebung und Optimieren der Datenbankleistung*.

Verbessern der Leistung mithilfe von Clusterindizes

Im Allgemeinen bleibt die Clusterbildung effektiver erhalten, wenn der Clusterindex eindeutig ist.

Unterschiede zwischen Integritätsbedingungen über Primärschlüssel bzw. eindeutige Schlüssel und eindeutigen Indizes

Es ist wichtig, zu verstehen, dass es keinen bedeutenden Unterschied zwischen einer Integritätsbedingung über Primärschlüssel bzw. eindeutige Schlüssel und einem eindeutigen Index gibt. Der Datenbankmanager verwendet eine Kombination aus einem eindeutigen Index und der Integritätsbedingung NOT NULL, um das Konzept relationaler Datenbanken von Integritätsbedingungen über Primärschlüssel bzw. eindeutige Schlüssel zu implementieren. Aus diesem Grund setzen eindeutige Indizes Integritätsbedingungen über Primärschlüssel nicht selbst um, denn sie lassen Nullwerte zu. (Zwar stellen Nullwerte unbekannte Werte dar, bei einer Indexierung wird ein Nullwert jedoch so behandelt, als würde er anderen Nullwerten entsprechen.)

Aus diesem Grund ist nur ein einziger Nullwert zulässig, wenn ein eindeutiger Index aus nur einer Spalte besteht. Ein weiterer Nullwert würde gegen die eindeutige Integritätsbedingung verstoßen. Bei einem eindeutigen Index aus mehreren Spalten kann dementsprechend eine bestimmte Kombination aus Werten und Nullen nur einmal verwendet werden.

Bidirektionale Indizes

Standardmäßig lassen bidirektionale Indizes Suchoperationen in beide Richtungen, d. h. vorwärts und rückwärts, zu. Mit der Klausel ALLOW REVERSE SCANS in der Anweisung CREATE INDEX ist eine vorwärts- und rückwärtsgerichtete Indexsuche möglich, d. h., der Index kann in der Reihenfolge, die bei der Erstellung des Index definiert wurde, und in umgekehrter Richtung durchsucht werden. Diese Option bietet folgende Möglichkeiten:

- Sie können die Funktionen MIN und MAX besser nutzen.
- Sie können vorherige Schlüssel abrufen.
- Der Datenbankmanager muss keine temporäre Tabelle mehr für die Rückwärtssuche erstellen.
- Redundante Indizes in umgekehrter Reihenfolge werden eliminiert.

Wenn DISALLOW REVERSE SCANS angegeben ist, kann der Index nicht rückwärts durchsucht werden. (Physisch wird er jedoch genau so erstellt, wie ein Index mit ALLOW REVERSE SCANS.)

Partitionierte und nicht partitionierte Indizes

Für partitionierte Daten können Indizes, die nicht partitioniert sind und sich in einem Tabellenbereich innerhalb einer Datenbankpartition befinden, Indizes, die

selbst über einen oder mehrere Tabellenbereiche innerhalb einer Datenbankpartition partitioniert sind, oder eine Kombination dieser beiden Arten von Indizes vorhanden sein. Partitionierte Indizes sind insbesondere dann von Vorteil, wenn Rollin-Operationen mit partitionierten Tabellen ausgeführt werden. (Dabei wird eine Datenpartition mithilfe der Klausel ATTACH PARTITION der Anweisung ALTER TABLE einer anderen Tabelle zugeordnet.)

Indizes für partitionierte Tabellen

Für partitionierte Tabellen können Indizes, die nicht partitioniert sind (in einem Tabellenbereich innerhalb einer Datenbankpartition), Indizes, die selbst über einen oder mehrere Tabellenbereiche innerhalb einer Datenbankpartition partitioniert sind, oder eine Kombination dieser beiden Arten von Indizes vorhanden sein.

Partitionierte Indizes bieten Vorteile, wenn Rollin-Operationen mit partitionierten Tabellen ausgeführt werden (dabei wird eine Datenpartition mithilfe der Klausel ATTACH PARTITION der Anweisung ALTER TABLE einer anderen Tabelle zugeordnet). Mit einem partitionierten Index können Sie die Indexpflege vermeiden, die andernfalls für nicht partitionierte Indizes ausgeführt werden müsste. Wenn eine partitionierte Tabelle einen nicht partitionierten Index hat, müssen Sie die Indexschlüssel aus den neu zugeordneten Partitionen mit der Anweisung SET INTEGRITY verwalten. Dies ist nicht nur zeitaufwändig, sondern kann auch je nach Anzahl der Zeilen, die durch das Rollin eingefügt werden, eine große Menge an Protokollspeicherplatz erfordern.

Es gibt einige Typen von Indizes, die nicht partitioniert werden können:

- Indizes zu nicht partitionierten Daten
- Indizes zu räumlichen Daten
- XML-Spaltenpfadindizes (vom System generiert)

Sie müssen diese Indizes immer als nicht partitionierte Indizes erstellen. Darüber hinaus muss der Indexschlüssel für partitionierte eindeutige Indizes alle Spalten aus dem Tabellenpartitionierungsschlüssel enthalten, und zwar unabhängig davon, ob sie von einem Benutzer oder vom System generiert wurden. Das Letztere wäre der Fall für Indizes, die vom System generiert werden, um Integritätsbedingungen für Eindeutigkeit oder Primärschlüssel für Daten umzusetzen.

Ab DB2 Version 9.7 Fixpack 1 können Sie einen Index zu XML-Daten für eine partitionierte Tabelle entweder als partitionierten oder als nicht partitionierten Index erstellen. Standardmäßig wird ein partitionierter Index erstellt. Eindeutige Indizes zu XML-Daten sind immer nicht partitioniert.

Nicht partitionierte Indizes für partitionierte Tabellen

Ein *nicht partitionierter Index* ist ein einzelnes Indexobjekt, das sich auf alle Zeilen in einer partitionierten Tabelle bezieht. Nicht partitionierte Indizes werden immer als unabhängige Indexobjekte in nur einem Tabellenbereich erstellt, selbst wenn sich die Tabellendatenpartitionen über mehrere Tabellenbereiche erstrecken.

Wenn Sie einen Index für eine partitionierte Tabelle erstellen, ist der Index standardmäßig ein *partitionierter Index*, sofern Sie nicht einen der folgenden Typen von Indizes erstellen:

- Einen eindeutigen Index, bei dem der Indexschlüssel nicht alle Tabellenpartitionierungsspalten umfasst
- Einen räumlichen Index

In diesen Fällen ist der Index, den Sie erstellen, ein nicht partitionierter Index. Es gibt jedoch Fälle, in denen es nützlich oder erforderlich ist, einen nicht partitionierten Index zu erstellen, auch wenn die Daten partitioniert sind. In diesen Fällen verwenden Sie die Klausel NOT PARTITIONED der Anweisung CREATE INDEX, um einen nicht partitionierten Index für eine partitionierte Tabelle zu erstellen. Wenn Sie einen nicht partitionierten Index erstellen, wird er standardmäßig im selben Tabellenbereich wie die erste sichtbare bzw. zugeordnete Datenpartition gespeichert. Abb. 43 zeigt ein Beispiel eines einzelnen Index X1, der auf alle Partitionen in einer Tabelle verweist. Der Index wurde im selben Tabellenbereich wie die erste sichtbare Partition für die Tabelle erstellt.

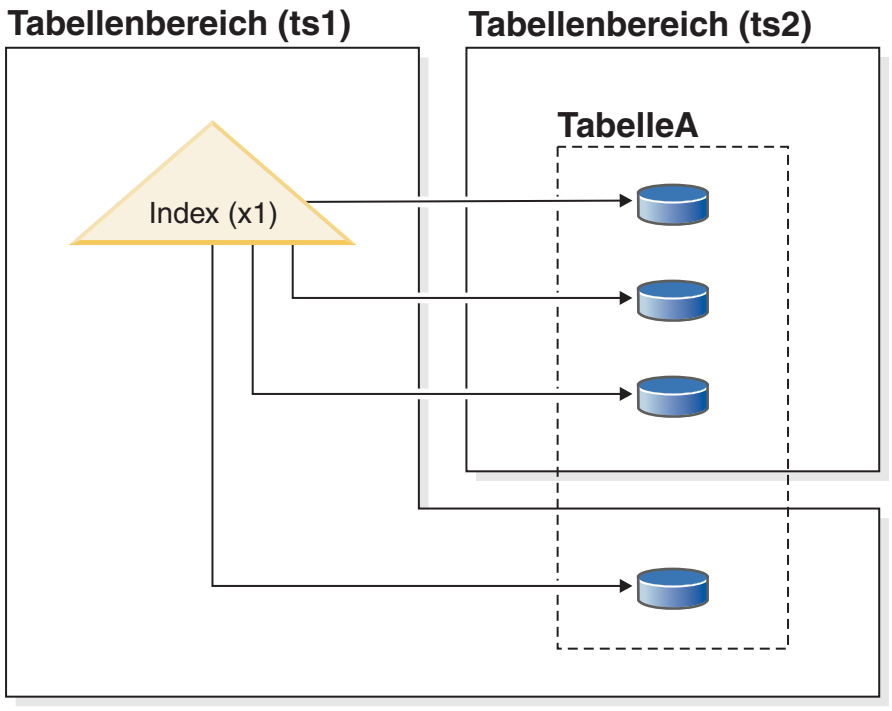


Abbildung 43. Nicht partitionierter Index für eine partitionierte Tabelle

Abb. 44 auf Seite 482 zeigt ein Beispiel für zwei nicht partitionierte Indizes. In diesem Fall befindet sich jede Indexpartition in einem Tabellenbereich, der von dem Tabellenbereich der Datenpartitionen separat ist. Beachten Sie wiederum, wie jeder Index auf alle Partitionen in der Tabelle verweist.

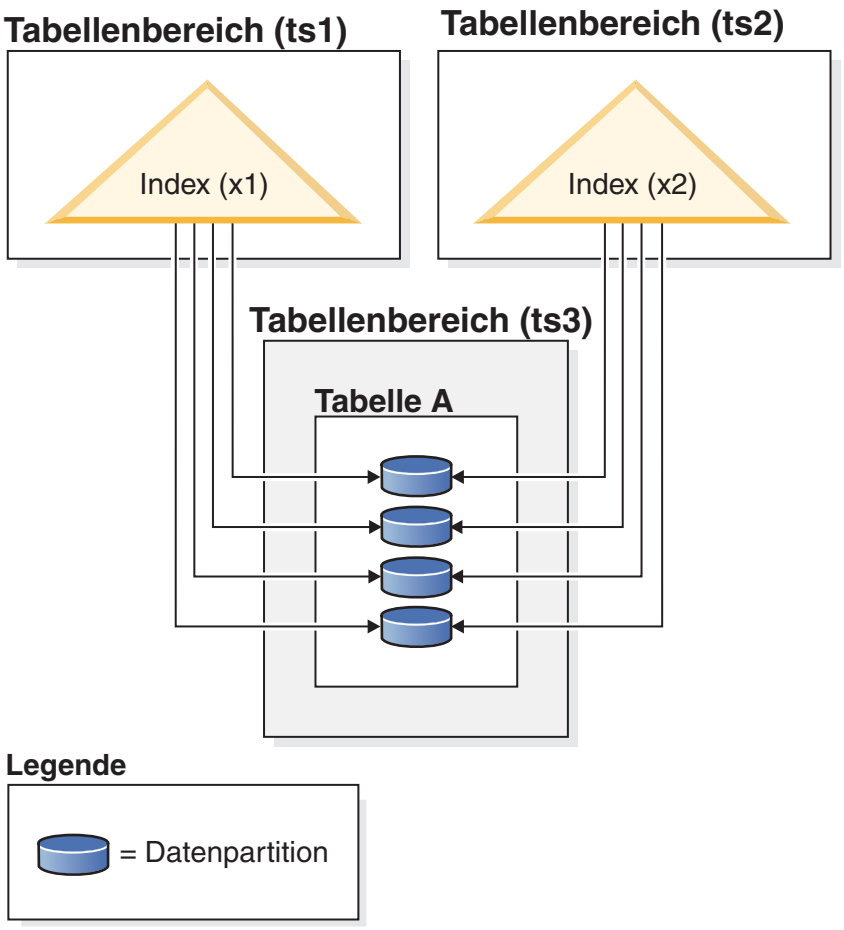


Abbildung 44. Nicht partitionierte Indizes für eine partitionierte Tabelle mit Indizes in eigenen Tabellenbereichen

Sie können die Position für einen nicht partitionierten Index zu folgenden Zeitpunkten überschreiben:

- Wenn Sie die Tabelle erstellen, indem Sie die Klausel INDEX IN der Anweisung CREATE TABLE verwenden.
- Wenn Sie den Index erstellen, indem Sie die Klausel IN der Anweisung CREATE INDEX verwenden.

Die zweite Methode hat immer Vorrang vor der ersten.

Wenn Sie Daten mit der Klausel ATTACH PARTITION in der Anweisung ALTER TABLE in eine partitionierte Tabelle einlagern (Rollin), müssen Sie die Anweisung SET INTEGRITY ausführen, um die Daten der zugeordneten Partition für Abfragen online verfügbar zu machen. Wenn die Indizes nicht partitioniert sind, kann diese Onlineverfügbarmachung der Daten der zugeordneten Partition eine zeitaufwendige Operation sein, die beträchtlichen Protokollspeicher benötigt, da die Anweisung SET INTEGRITY Daten aus der neu zugeordneten Partition in die nicht partitionierten Indizes einfügen muss.

Die Anweisung SET INTEGRITY muss nach der Aufhebung der Zuordnung einer Partition nicht ausgeführt werden.

Partitionierte Indizes für partitionierte Tabellen

Ein *partitionierter Index* besteht aus einer Reihe von *Indexpartitionen*, die jeweils die Indexeinträge für eine einzelne Datenpartition enthalten. Jede Indexpartition enthält Verweise nur auf die Daten in der entsprechenden Datenpartition. Sowohl vom System als auch vom Benutzer generierte Indizes können partitioniert sein.

Ein partitionierter Index ist in folgenden Situationen von Vorteil:

- Sie lagern Daten in partitionierte Tabellen mit der Klausel ATTACH PARTITION der Anweisung ALTER TABLE ein (Rollin) oder aus partitionierten Tabellen mit der Klausel DETACH PARTITION derselben Anweisung aus (Rollout). Bei einem nicht partitionierten Index kann die Anweisung SET INTEGRITY, die ausgeführt werden muss, bevor die neu zugeordnete Partition verfügbar ist, viel Zeit und große Mengen an Protokollspeicher beanspruchen. Wenn Sie eine Tabellenpartition zuordnen (ATTACH), die einen partitionierten Index verwendet, müssen Sie die Anweisung SET INTEGRITY, die Tasks wie Bereichsprüfungen und Prüfungen von Integritätsbedingungen ausführt, immer noch ausführen.

  Tipp: Wenn die Prüfung der Datenintegrität (einschließlich Bereichsprüfung und Prüfung anderer Integritätsbedingungen) durch vom Datenserver unabhängige Anwendungslogik vor einer Zuordnungsoperation erfolgen kann, können die neu zugeordneten Daten deutlich früher verfügbar gemacht werden. Sie können den Prozess für Dateneinlagerung optimieren, indem Sie mithilfe der Anweisung SET INTEGRITY...ALL IMMEDIATE UNCHECKED die Bereichsprüfung und die Prüfung auf ungültige Integritätsbedingungen überspringen. In diesem Fall verlässt die Tabelle den Status 'Festlegen der Integrität anstehend' und die neuen Daten sind sofort für Anwendungen verfügbar, sofern die Zieltabelle keine nicht partitionierten Benutzerindizes enthält.

  Wenn die Indizes für die Quellentabelle mit den Indexpartitionen für die Zieltabelle, beeinträchtigt die Verarbeitung der Anweisung SET INTEGRITY nicht die Leistung oder die Protokollverarbeitung, die mit einer Indexwartung verbunden ist. Die neu eingelagerten Daten stehen schneller als bei nicht partitionierten Indizes zur Verfügung. Weitere Informationen zum Indexabgleich finden Sie im Abschnitt „Bedingungen für den Abgleich eines Quellentabellenindex mit einem partitionierten Zieltabellenindex bei der Ausführung von ATTACH PARTITION" im Handbuch *Partitionierung und Clustering*.
- Sie führen Wartungstasks an Daten in einer bestimmten Partition aus, die eine Indexreorganisation erfordern. Betrachten Sie zum Beispiel eine Tabelle mit 12 Partitionen, die jeweils einem bestimmten Monat des Jahres entsprechen. Möglicherweise möchten Sie viele Zeilen, die für einen bestimmten Monat des Jahres spezifisch sind, aktualisieren oder löschen. Diese Aktion könnte zu einer Fragmentierung des Index führen, die dazu führt, dass eine Indexreorganisation erforderlich wird. Bei einem partitionierten Index haben Sie die Möglichkeit, nur die Indexpartition zu reorganisieren, die der Datenpartition entspricht, in der die Änderungen ausgeführt wurden. Dadurch lassen sich gegenüber einer Reorganisation eines gesamten, nicht partitionierten Index erhebliche Zeiteinsparungen erzielen.

Es gibt einige Typen von Indizes, die nicht partitioniert werden können:

- Indizes zu nicht partitionierten Daten
- Indizes zu räumlichen Daten
- XML-Spaltenpfadindizes (vom System generiert)

Sie müssen diese Indizes immer als nicht partitionierte Indizes erstellen. Darüber hinaus muss der Indexschlüssel für partitionierte eindeutige Indizes alle Spalten

aus dem Tabellenpartitionierungsschlüssel enthalten, und zwar unabhängig davon, ob sie von einem Benutzer oder vom System generiert wurden. Das Letztere wäre der Fall für Indizes, die vom System generiert werden, um Integritätsbedingungen für Eindeutigkeit oder Primärschlüssel für Daten umzusetzen.

Abb. 45 zeigt ein Beispiel für partitionierte Indizes.

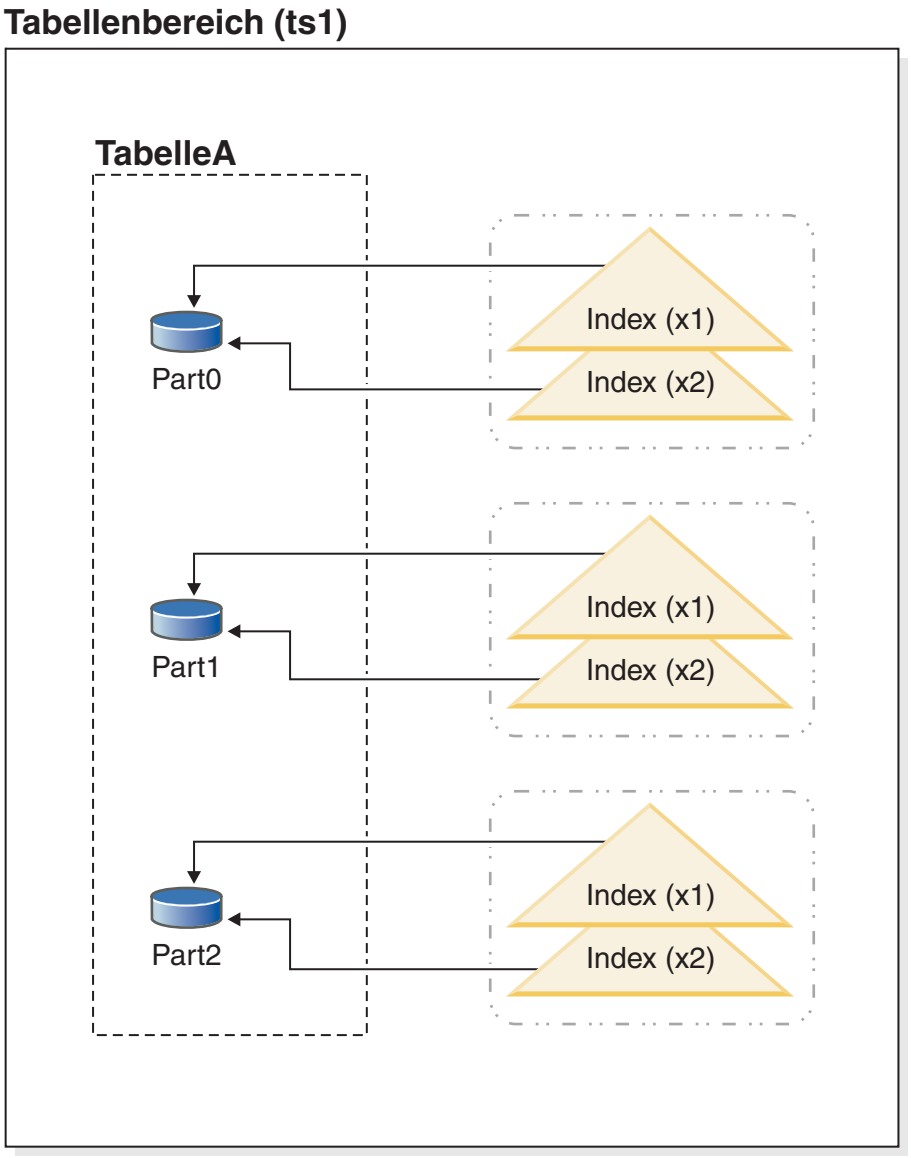


Abbildung 45. Partitionierte Indizes, die einen Tabellenbereich mit Datenpartitionen einer Tabelle gemeinsam nutzen

In diesem Beispiel befinden sich alle Datenpartitionen für Tabelle A und alle Indexpartitionen für Tabelle A in einem einzigen Tabellenbereich. Die Indexpartitionen verweisen nur auf die Zeilen in der Datenpartition, der sie zugeordnet sind. (Dies unterscheidet einen partitionierten Index von einem nicht partitionierten Index, der auf *alle* Zeilen in *allen* Datenpartitionen verweist.) Darüber hinaus befinden sich die Indexpartitionen für eine Datenpartition im selben Indexobjekt. Diese spezielle Anordnung von Indizes und Indexpartitionen wird zum Beispiel durch Anweisungen wie die folgenden eingerichtet:

```
CREATE TABLE A (spalten) in ts1
      PARTITION BY RANGE (spaltenausdruck)
      (PARTITION PART0 STARTING FROM konstante ENDING konstante,
       PARTITION PART1 STARTING FROM konstante ENDING konstante,
       PARTITION PART2 STARTING FROM konstante ENDING konstante,

   CREATE INDEX x1 ON A (...) PARTITIONED;
   CREATE INDEX x2 ON A (...) PARTITIONED;
```

Abb. 46 zeigt ein weiteres Beispiel für einen partitionierten Index.

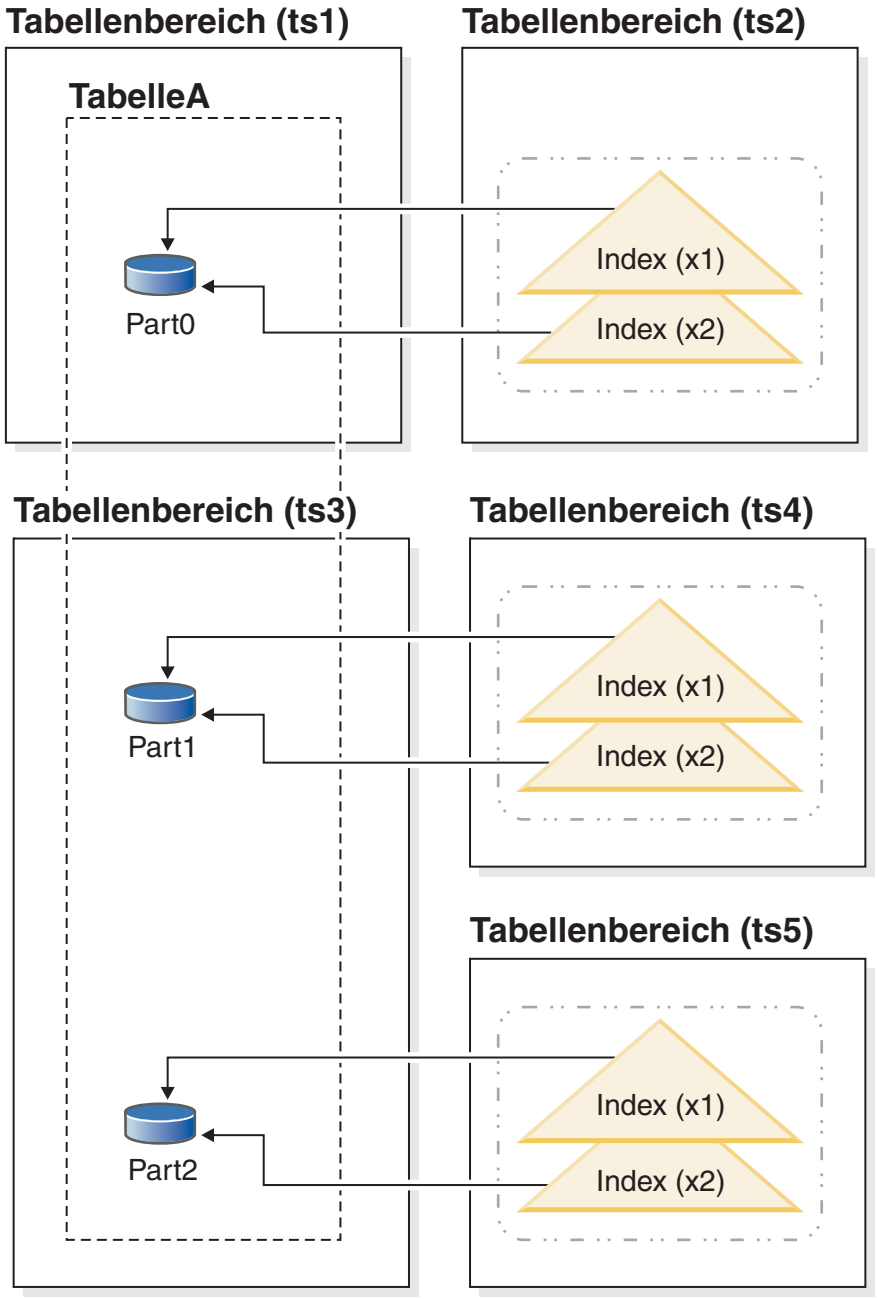


Abbildung 46. Partitionierte Indizes mit Datenpartitionen und Indexpartitionen in verschiedenen Tabellenbereichen

In diesem Beispiel sind die Datenpartitionen für Tabelle A auf die zwei Tabellenbereiche TS1 und TS3 verteilt. Die Indexpartitionen befinden sich ebenfalls in verschiedenen Tabellenbereichen. Die Indexpartitionen verweisen nur auf die Zeilen in der Datenpartition, der sie zugeordnet sind. Diese spezielle Anordnung von Indizes und Indexpartitionen wird zum Beispiel durch Anweisungen wie die folgenden eingerichtet:

```
CREATE TABLE A (spalten)
      PARTITION BY RANGE (spaltenausdruck)
      (PARTITION PART0 STARTING FROM konstante ENDING konstante IN ts1 INDEX IN ts2,
       PARTITION PART1 STARTING FROM konstante ENDING konstante IN ts3 INDEX IN ts4,
       PARTITION PART2 STARTING FROM konstante ENDING konstante IN ts3,INDEX IN ts5)

   CREATE INDEX x1 ON A (...);
   CREATE INDEX x2 ON A (...);
```

In diesem Fall wurde die Klausel PARTITIONED in der Anweisung CREATE INDEX nicht angegeben. Die Indizes werden trotzdem als partitionierte Indizes erstellt, da diese Einstellung das Standardverhalten für partionierte Tabellen ist.

Das Diagramm in Abb. 47 auf Seite 487 zeigt ein Beispiel für eine partitionierte Tabelle mit nicht partitionierten und partitionierten Indizes.

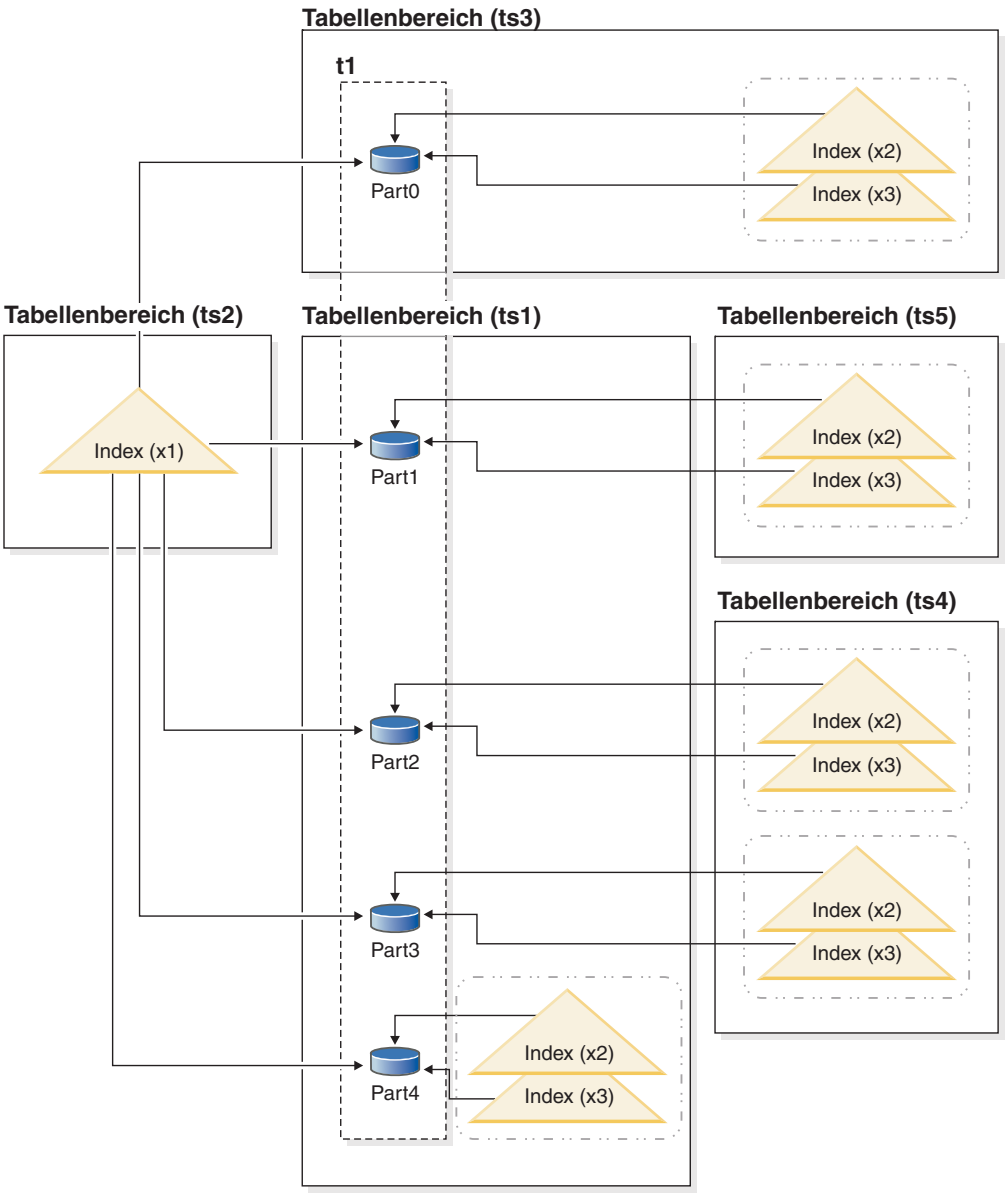


Abbildung 47. Kombination aus nicht partitionierten und partitionierten Indizes für eine partitionierte Tabelle

In diesem Diagramm ist der Index X1 ein nicht partitionierter Index, der auf alle Partitionen der Tabelle T1 verweist. Die Indizes X2 und X3 sind partitionierte Indizes, die sich in verschiedenen Tabellenbereichen befinden. Diese spezielle Anordnung von Indizes und Indexpartitionen wird zum Beispiel durch Anweisungen wie die folgenden eingerichtet:

```
CREATE TABLE t1 (spalten) in ts1 INDEX IN ts2 1
   PARTITION BY RANGE (spaltenausdruck)
   (PARTITION PART0 STARTING FROM konstante ENDING konstante  IN ts3, 2
    PARTITION PART1 STARTING FROM konstante ENDING konstante  INDEX IN ts5,
    PARTITION PART2 STARTING FROM konstante ENDING konstante  INDEX IN ts4,
    PARTITION PART3 STARTING FROM konstante ENDING konstante  INDEX IN ts4,
    PARTITION PART4 STARTING FROM konstante ENDING konstante)

CREATE INDEX x1 ON t1 (...) NOT PARTITIONED;
CREATE INDEX x2 ON t1 (...) PARTITIONED;
CREATE INDEX x3 ON t1 (...) PARTITIONED;
```

Beachten Sie Folgendes:

- Der nicht partitionierte Index X1 wird in Tabellenbereich TS2 gespeichert, weil diese Position der für nicht partitionierte Indizes für Tabelle T1 angegebene Standard ist (siehe **1**).
- Die Indexpartition für Datenpartition 0 (Part0) wird in Tabellenbereich TS3 gespeichert, weil die Standardposition für eine Indexpartition dieselbe ist, wie die der Datenpartition, auf die sie verweist (siehe **2**).
- Die Partition Part4 wird in TS1 gespeichert, dem Standardtabellenbereich für Datenpartitionen in Tabelle T1 (siehe **1**). Die Indexpartitionen für diese Datenpartition befinden sich ebenfalls in TS1, wiederum weil die Standardposition für eine Indexpartition dieselbe ist, wie die der Datenpartition, auf die sie verweist.

Wichtig: Im Unterschied zu nicht partitionierten Indizes können Sie für partitionierte Indizes die Klausel INDEX IN der Anweisung CREATE INDEX nicht verwenden, um den Tabellenbereich anzugeben, in dem Indexpartitionen gespeichert werden sollen. Die einzige Möglichkeit, die Standardposition für Indexpartitionen zu ändern, besteht darin, die Position bei der Erstellung der Tabelle mithilfe der Klausel INDEX IN *auf Partitionsebene* der Anweisung CREATE TABLE anzugeben. Die Klausel INDEX IN auf Tabellenebene hat keine Wirkung auf die Positionierung von Indexpartitionen.

Sie erstellen partitionierte Indizes für eine partitionierte Tabelle, indem Sie die Option PARTITIONED in einer Anweisung CREATE INDEX angeben. Für eine Tabelle mit dem Namen SALES, die über die Spalte `sales_date` als Tabellenpartitionierungsschlüssel partitioniert wird, könnten Sie einen partitionierten Index zum Beispiel mithilfe einer Anweisung wie der folgenden erstellen:

```
CREATE INDEX partIDbydate on SALES (sales_date, partID) PARTITIONED
```

Wenn Sie einen eindeutigen partitionierten Index erstellen, müssen die Tabellenpartitionierungsspalten in den Indexschlüsselspalten enthalten sein. Nehmen Sie an, im obigen Beispiel soll ein partitionierter Index mit der folgenden Anweisung erstellt werden:

```
CREATE UNIQUE INDEX uPartID on SALES (partID) PARTITIONED
```

In diesem Fall würde die Ausführung der Anweisung fehlschlagen, weil die Spalte `sales_date`, die den Tabellenpartitionierungsschlüssel bildet, nicht im Indexschlüssel enthalten wäre.

Wenn Sie bei der Erstellung eines Index für eine partitionierte Tabelle das Schlüsselwort PARTITIONED nicht angeben, erstellt der Datenbankmanager standardmäßig einen partitionierten Index, sofern nicht eine der folgenden Bedingungen erfüllt ist:

- Sie erstellen einen eindeutigen Index und der Indexschlüssel enthält nicht alle Tabellenpartitionierungsschlüssel.
- Sie erstellen einen der Typen von Indizes, die in der Beschreibung zu Anfang dieses Abschnitts als Indizes, die nicht als partitionierte Indizes erstellt werden können, aufgeführt sind.

In beiden Fällen wird der Index als nicht partitionierter Index erstellt.

Obwohl bei der Erstellung eines nicht partitionierten Index mit einer Definition, die mit der eines vorhandenen nicht partitionierten Index übereinstimmt, SQL0605W zurückgegeben wird, kann ein partitionierter Index mit einem nicht partitionierten Index mit ähnlicher Definition koexistieren. Diese Koexistenz soll die Einführung partitionierter Indizes vereinfachen.

Entwerfen von Indizes

In der Regel dienen Indizes zur Beschleunigung des Zugriffs auf eine Tabelle. Sie können aber auch Zwecke des logischen Datenentwurfs erfüllen.

Zum Beispiel lässt ein eindeutiger Index nicht zu, dass in die Spalten ein Wert mehrfach eingegeben wird, wodurch gewährleistet wird, dass nicht zwei Zeilen in einer Tabelle identisch sind. Indizes können auch erstellt werden, um die Werte in einer Spalte aufsteigend oder absteigend anzuordnen.

Wichtig: Bei der Erstellung von Indizes sollten Sie beachten, dass sie zwar die Leseleistung verbessern können, sich jedoch negativ auf die Schreibleistung auswirken. Der Grund für diese negative Auswirkung ist, dass für jede Zeile, die der Datenbankmanager in eine Tabelle schreibt, auch die betroffenen Indizes aktualisiert werden müssen. Erstellen Sie Indizes deshalb nur, wenn sie einen eindeutigen Vorteil für die Gesamtleistung bringen.

Berücksichtigen Sie bei der Erstellung von Indizes außerdem die Struktur der Tabellen und den Typ von Abfragen, die am häufigsten für sie durchgeführt werden. So sind beispielsweise Spalten, die in der Klausel WHERE einer häufig abgesetzten Abfrage auftreten, geeignete Kandidaten für Indizes. Bei weniger häufig ausgeführten Abfragen kann der Aufwand, der für einen Index für die Leistung in Anweisungen INSERT und UPDATE entsteht, die Vorteile zunichte machen.

In ähnlicher Weise können Spalten, die in einer GROUP BY-Klausel einer häufigen Abfrage auftreten, die Erstellung eines Index vorteilhaft nutzen, insbesondere dann, wenn die Anzahl der zum Gruppieren der Zeilen verwendeten Werte im Gegensatz zur Anzahl der gruppierten Zeilen gering ist.

Beachten Sie bei der Erstellung von Indizes, dass diese auch komprimiert werden können. Sie können die Indizes später ändern, indem Sie mithilfe der Anweisung ALTER INDEX die Komprimierung aktivieren oder inaktivieren.

Zum Entfernen oder Löschen von Indizes können Sie den Befehl DROP INDEX verwenden. Das Löschen von Indizes hat die umgekehrten Anforderungen des Einfügens von Indizes; das heißt, die Indexeinträge müssen entfernt (bzw. als gelöscht markiert) werden.

Richtlinien und Aspekte für den Entwurf von Indizes

- Obwohl die Reihenfolge der Spalten, die einen Indexschlüssel bilden, bei der Erstellung des Indexschlüssels keine Rolle spielt, kann sie für das Optimierungsprogramm bei der Entscheidung von Bedeutung sein, ob ein Index verwendet werden soll. Wenn eine Abfrage zum Beispiel eine Klausel `ORDER BY col1,col2` enthält, könnte ein Index, der für die Spalten `(col1,col2)` erstellt wurde, verwendet werden, während ein Index, der für die Spalten `(col2,col1)` erstellt wurde, keine Hilfe darstellt. Ähnlich gilt, dass für eine Abfrage, die mit einer Bedingung wie `where col1 >= 50 and col1 <= 100` oder `where col1=74` geschrieben wird, ein Index für die Spalte `(col1)` oder für die Spalten `(col1,col2)` hilfreich sein könnte, während ein Index für die Spalten `(col2,col1)` wesentlich weniger nützlich ist.

  Anmerkung: Ordnen Sie nach Möglichkeit die Spalten in einem Indexschlüssel immer von der distinktivsten zur am wenigsten distinktiven Spalte. Diese Anordnung bietet die beste Leistung.

- Für eine bestimmte Tabelle kann eine beliebige Anzahl von Indizes definiert werden (maximal 32.767), und diese Indizes können sich positiv auf die Leistung von Abfragen auswirken. Der Indexmanager muss die Indizes bei Aktualisierungs-, Lösch- und Einfügeoperationen pflegen. Daher kann die Erstellung einer großen Anzahl von Indizes für eine Tabelle, die häufig aktualisiert wird, die Verarbeitung von Anforderungen verlangsamen. Durch umfangreiche Indexschlüssel kann es ebenfalls zu einer Verlangsamung der Verarbeitungsgeschwindigkeit für Anforderungen kommen. Die Verwendung von Indizes ist also nur dann sinnvoll, wenn sich klare Vorteile für den häufigen Zugriff ergeben.
- Spaltendaten, die nicht Teil des eindeutigen Indexschlüssels sind, jedoch im Index gespeichert oder gepflegt werden sollen, werden als INCLUDE-Spalten bezeichnet. INCLUDE-Spalten können nur für eindeutige Indizes angegeben werden. Bei der Erstellung eines Index mit INCLUDE-Spalten werden nur die eindeutigen Schlüsselspalten sortiert und im Hinblick auf Eindeutigkeit geprüft. Die Verwendung von INCLUDE-Spalten kann einen reinen Indexzugriff für den Datenabruf ermöglichen, was zu einer Verbesserung der Leistung führt.
- Wenn die Tabelle, für die ein Index erstellt wird, leer ist, wird der Index zwar erstellt, jedoch werden erst Indexeinträge erstellt, wenn die Tabelle geladen oder Zeilen eingefügt werden. Ist die Tabelle nicht leer, erstellt der Datenbankmanager die Indexeinträge während der Verarbeitung der Anweisung CREATE INDEX.
- Bei einem *Clusterindex* versucht der Datenbankmanager, die neuen Zeilen für die Tabelle physisch nahe bei vorhandenen Zeilen mit ähnlichen Schlüsselwerten (wie durch den Index definiert) einzufügen.
- Wenn Sie einen *Primärschlüsselindex* als Clusterindex verwenden wollen, sollte in der Anweisung CREATE TABLE kein Primärschlüssel angegeben werden. Wenn der Primärschlüssel einmal erstellt ist, kann der zugehörige Index nicht geändert werden. Setzen Sie stattdessen die Anweisung CREATE TABLE ohne Primärschlüsselklausel (PRIMARY KEY) ab. Führen Sie anschließend die Anweisung CREATE INDEX aus, in der Sie die Clustering-Attribute angeben. Verwenden Sie schließlich die Anweisung ALTER TABLE, um einen Primärschlüssel hinzuzufügen, der dem gerade erstellten Index entspricht. Dieser Index wird als Primärschlüsselindex verwendet.
- Wenn Sie eine *partitionierte Tabelle* haben, ist standardmäßig jeder Index, den Sie für die Tabelle erstellen, ein *partitionierter Index*, sofern Sie keinen eindeutigen Index erstellen, der nicht den Partitionierungsschlüssel enthält. Sie können den Index auch als *nicht partitionierten Index* erstellen.

  Ab DB2 Version 9.7 Fixpack 1 können Sie einen Index zu XML-Daten für eine partitionierte Tabelle entweder als partitionierten oder als nicht partitionierten Index erstellen. Standardmäßig wird ein partitionierter Index erstellt.

  Partitionierte Indizes bieten Vorteile, wenn Rollin-Operationen mit partitionierten Tabellen ausgeführt werden (dabei wird eine Datenpartition mithilfe der Klausel ATTACH PARTITION der Anweisung ALTER TABLE einer anderen Tabelle zugeordnet). Mit einem partitionierten Index können Sie die Indexpflege vermeiden, die andernfalls für nicht partitionierte Indizes ausgeführt werden müsste. Wenn eine partitionierte Tabelle einen nicht partitionierten Index hat, müssen Sie die Indexschlüssel aus den neu zugeordneten Partitionen mit der Anweisung SET INTEGRITY verwalten. Dies ist nicht nur zeitaufwändig, sondern kann auch je nach Anzahl der Zeilen, die durch das Rollin eingefügt werden, eine große Menge an Protokollspeicherplatz erfordern.
- Indizes belegen Plattenspeicherplatz. Die Menge an Plattenspeicherplatz variiert in Abhängigkeit von der Länge der Schlüsselspalten und der Anzahl der indexierten Zeilen. Die Größe des Index nimmt zu, je mehr Daten in die Tabelle eingefügt werden. Aus diesem Grund müssen Sie den Umfang der zu indexieren-

den Daten bei der Planung der Größe der Datenbank berücksichtigen. Im Folgenden sind einige der Aspekte für die Dimensionierung des Index aufgeführt:

- Integritätsbedingungen über Primärschlüssel und eindeutige Schlüssel führen immer zur Erstellung eines systemgenerierten eindeutigen Index.
- Die Erstellung einer MDC- oder ITC-Tabelle führt ebenfalls zur Generierung von Blockindizes durch das System.
- XML-Spalten haben immer die Erstellung systemgenerierter Indizes, einschließlich Spaltenpfadindizes und Regionsindizes, zur Folge.
- Es ist in der Regel von Vorteil, Indizes für Spalten mit Integritätsbedingungen über Fremdschlüssel zu erstellen.
- Ob der Index komprimiert wird oder nicht (mit der Option COMPRESS).

Anmerkung: Die maximale Anzahl der Spalten in einem Index beträgt 64. Jedoch beträgt die maximale Anzahl an Spalten in einem Index beim Indexieren einer typisierten Tabelle 63. Die maximale Länge eines Indexschlüssels einschließlich aller Komponenten beträgt *IndexPageSize* ÷ 4. Für eine Tabelle sind maximal 32.767 Indizes zulässig. Die maximale Länge eines Indexschlüssels darf die für die Seitengröße geltende Längenbegrenzung für Indexschlüssel nicht überschreiten. Informationen zu gespeicherten Längen finden Sie im Abschnitt zur „Anweisung CREATE TABLE". Informationen zu Schlüssellängenbegrenzungen finden Sie im Abschnitt „SQL- und XQuery-Begrenzungen".

- Bei einem Datenbankupgrade werden vorhandene Indizes nicht komprimiert. Wenn für eine Tabelle die Datenzeilenkomprimierung aktiviert ist, werden neue Indizes nach dem Upgrade komprimiert, sofern in der Anweisung CREATE INDEX nicht die Option COMPRESS NO angegeben wird.

Tools für das Entwerfen von Indizes

Sobald Sie Ihre Tabellen erstellt haben, müssen Sie überlegen, wie schnell der Datenbankmanager Daten aus den Tabellen abrufen kann. Sie können den Designadvisor oder den Befehl **db2advis** zur Unterstützung beim Entwerfen Ihrer Indizes verwenden.

Das Erstellen nützlicher Indizes für Ihre Tabellen kann zur deutlichen Verbesserung der Abfrageleistung beitragen. Ähnlich wie bei Buchindizes können mithilfe von Tabellenindizes bestimmte Informationen mit einem minimalen Suchaufwand schnell gefunden werden. Die Verwendung eines Index zum Abrufen bestimmter Zeilen aus einer Tabelle kann die Anzahl aufwandsintensiver Ein-/Ausgabeoperationen reduzieren, die der Datenbankmanager durchführen muss. Der Grund hierfür ist, dass es ein Index dem Datenbankmanager ermöglicht, durch Lesen einer relativ geringen Anzahl von Datenseiten nach einer Zeile zu suchen; er muss keine aufwandsintensive Suche für sämtliche Datenseiten betreiben, bis alle Übereinstimmungen gefunden sind.

Der Designadvisor von DB2 ist ein Tool, das Ihnen helfen kann, die Auslastungsleistung Ihres Systems erheblich zu verbessern. Die Aufgabe, zu entscheiden, welche Indizes, MQTs (MQT - Materialized Query Table, gespeicherte Abfragetabelle), Clusteringdimensionen oder Datenbankpartitionen für eine komplexe Auslastung zu erstellen sind, kann sich als äußerst schwierig gestalten. Der Designadvisor ermittelt alle Objekte, die zur Verbesserung der Leistung Ihrer Auslastung erforderlich sind. Für einen gegebenen Satz von SQL-Anweisungen in einer Auslastung generiert der Designadvisor Empfehlungen für folgende Objekte und Maßnahmen:

- Neue Indizes

- Neue gespeicherte MQTs
- Umwandlung in MDC-Tabellen (mit mehrdimensionalem Clustering)
- Umverteilen von Tabellen
- Löschen von Indizes und MQTs, die durch die angegebene Auslastung nicht genutzt werden (über das GUI-Tool)

Über den Designadvisor können Sie einige oder alle dieser Empfehlungen unverzüglich implementieren oder ihre Implementierung für einen späteren Zeitpunkt terminieren.

Der Designadvisor kann die folgenden Aufgaben vereinfachen:

- Planen oder Einrichten einer neuen Datenbank
- Optimieren der Auslastungsleistung

Speicherbedarf für Indizes

Beim Entwerfen von Indizes müssen Sie sich mit dem anfallenden Speicherplatzbedarf vertraut machen. Für komprimierte Indizes können die Schätzwerte, die Sie aus den Formeln in diesem Abschnitt ableiten, als Obergrenze verwendet werden. Jedoch wird der Speicherbedarf wahrscheinlich wesentlich geringer sein.

Speicherbedarf für nicht komprimierte Indizes

Für jeden nicht komprimierten Index kann der erforderliche Speicherbereich wie folgt abgeschätzt werden:

(*durchschnittliche Indexschlüsselgröße* + *Indexschlüsselaufwand*) × *Anzahl der Zeilen* × 2

Dabei gilt Folgendes:

- Die *durchschnittliche Indexschlüsselgröße* ist die Bytezahl jeder Spalte im Indexschlüssel. Verwenden Sie bei der Abschätzung der durchschnittlichen Spaltengröße für VARCHAR- und VARGRAPHIC-Spalten einen Durchschnittswert der aktuellen Datengröße plus zwei Byte.
- Der *Indexschlüsselaufwand* hängt vom Typ der Tabelle ab, für die der Index erstellt wird:

Tabelle 77. Indexschlüsselaufwand für verschiedene Tabellen

Typ von Tabellenbereich	Tabellentyp	Indextyp	Indexschlüsselaufwand
Beliebig	Beliebig	XML-Pfade oder -Regionen	11 Byte
Regulär	Nicht partitioniert	Beliebig	9 Byte
	Partitioniert	Partitioniert	9
		Nicht partitioniert	11
Groß (Large)	Partitioniert	Partitioniert	11
		Nicht partitioniert	13

- Die *Anzahl der Zeilen* ist die Anzahl der Zeilen in einer Tabelle bzw. die Anzahl der Zeilen in einer bestimmten Datenpartition. Die Verwendung der Anzahl von Zeilen in der gesamten Tabelle in dieser Berechnung liefert einen Schätzwert für die Größe des Index (bei einem nicht partitionierten Index) bzw. für alle Index-

partitionen zusammen (bei einem partitionierten Index). Bei Verwendung der Anzahl von Zeilen in einer Datenpartition erhalten Sie einen Schätzwert für die Größe der Indexpartition.
- Der Faktor „2" ist für den Systemaufwand, z. B. für Nichtblattseiten und freien Speicherbereich.

Anmerkung:

1. Fügen Sie für jede Spalte, die Nullwerte zulässt, ein zusätzliches Byte für den Nullanzeiger hinzu.
2. Für Blockindizes, die intern für MDC-Tabellen (MDC = Multidimensional Clustering) oder ITC-Tabellen (ITC = Insert Time Clustering) erstellt werden, ist die „Anzahl der Zeilen" durch die „Anzahl der Blöcke" zu ersetzen.

Speicherbedarf für XML-Indizes

Für jeden Index für eine XML-Spalte kann der erforderliche Speicherbereich wie folgt abgeschätzt werden:

(*durchschnittlicher Indexschlüssel* + *Indexschlüsselaufwand*) × *Anzahl indexierter Knoten* × 2

Dabei gilt Folgendes:

- Der *durchschnittliche Indexschlüssel* ist die Summe der Schlüsselkomponenten, die den Index bilden. Der XML-Index setzt sich aus verschiedenen XML-Schlüsselkomponenten und einem Wert (SQL-Datentyp: sql-data-type) zusammen:

  14 + *variabler Aufwand* + *Bytezahl des SQL-Datentyps*

  Dabei gilt Folgendes:
  - 14 stellt die Anzahl der Byte für den festen Aufwand dar.
  - Der *variable Aufwand* ist die durchschnittliche Tiefe des indexierten Knotens plus 4 Byte.
  - Die *Bytezahl des SQL-Datentyps* (sql-data-type) folgt den gleichen Regeln wie SQL.
- Die *Anzahl indexierter Knoten* ist die Anzahl der einzufügenden Dokumente multipliziert mit der Anzahl von Knoten in einem Musterdokument, die dem XML-Musterausdruck (XMLPATTERN) in der Indexdefinition entsprechen. Die *Anzahl indexierter Knoten* kann die Anzahl der Knoten in einer Partition oder der gesamten Tabelle sein.

Temporärer Speicherbedarf für die Indexerstellung

Bei der Indexerstellung ist temporärer Speicherplatz erforderlich. Der maximal während der Indexerstellung erforderliche temporäre Speicherplatz kann folgendermaßen abgeschätzt werden:

(*durchschnittliche Indexschlüsselgröße* + *Indexschlüsselaufwand*) × *Anzahl der Zeilen* × 3,2

Für Indizes, für die mehr als ein Indexschlüssel pro Zeile vorhanden sein könnte, wie zum Beispiel räumliche Indizes, Indizes zu XML-Spalten und interne Indizes für XML-Regionen, kann der erforderliche temporäre Speicherbereich wie folgt abgeschätzt werden:

(*durchschnittliche Indexschlüsselgröße* + *Indexschlüsselaufwand*) × *Anzahl indexierter Knoten* × 3.2

Dabei wird der Faktor „3,2" für den Indexaufwand sowie für den Speicherbereich einkalkuliert, der für Sortierungen während der Indexerstellung erforderlich ist. Die *Anzahl der Zeilen* bzw. die *Anzahl indexierter Knoten* ist die Anzahl in einer gesamten Tabelle oder in einer bestimmten Datenpartition.

Anmerkung: Für nicht eindeutige Indizes wird nur eine Kopie eines bestimmten doppelten Schlüsseleintrags auf einem bestimmten Blattknoten gespeichert. Bei Indizes für Tabellen in großen Tabellenbereichen (Typ LARGE), ist die Größe für doppelte Schlüssel 9 für nicht partitionierte Indizes und 7 für partitionierte Indizes und Indizes für nicht partitionierte Tabellen. Für Indizes für Tabellen in regulären Tabellenbereichen (Typ REGULAR) sind diese Werte 7 für nicht partitionierte Indizes sowie 5 für partitionierte Indizes und Indizes für nicht partitionierte Tabellen. Die einzige Ausnahme von diesen Regeln gelten für Indizes zu XML-Pfaden und XML-Regionen, bei denen die Größe doppelter Schlüssel immer 7 ist. Der oben gezeigte Schätzwert geht davon aus, dass keine doppelten Schlüssel vorhanden sind. Der zum Speichern eines Index erforderliche Speicherbereich kann durch die oben gezeigte Formel eventuell zu groß abgeschätzt werden.

Wenn die Anzahl der Indexknoten ein Datenvolumen von 64 KB überschreitet, wird beim Einfügen ein temporärer Speicherbereich benötigt. Die Größe des temporären Speicherbereichs lässt sich wie folgt abschätzen:

durchschnittliche Indexschlüsselgröße × *Anzahl indexierter Knoten* × 1.2

Schätzen der Anzahl von Schlüsseln pro Blattseite

Die beiden folgenden Formeln können zum Abschätzen der Anzahl von Schlüsseln pro Indexblattseite verwendet werden. (Die zweite Formel liefert einen etwas genaueren Schätzwert.) Die Genauigkeit dieser Schätzwerte hängt im Wesentlichen davon ab, wie gut die verwendeten Durchschnittswerte die tatsächlichen Daten widerspiegeln.

Anmerkung: Für SMS-Tabellenbereiche entspricht der minimale Speicherbedarf für Blattseiten dem Dreifachen der Seitengröße. Für DMS-Tabellenbereiche entspricht der minimale Speicherbedarf einem EXTENTSIZE großen Speicherbereich.

1. Die Durchschnittszahl von Schlüsseln pro Blattseite kann mit folgender Formel grob abgeschätzt werden:

   $$((0{,}9 * (U - (M \times 2))) \times (D + 1)) \div (K + 7 + (Ds \times D))$$

   Dabei gilt Folgendes:

   - *U*, der verwendbare Speicherbereich auf einer Seite, entspricht ungefähr der Seitengröße minus 100. Bei einer Seitengröße von 4096 ist U zum Beispiel gleich 3996.
   - *M* = *U* ÷ (9 + *minimale Schlüsselgröße*)
   - *Ds* = *Größe doppelter Schlüssel* (Siehe Anmerkung unter „Temporärer Speicherbedarf für die Indexerstellung".)
   - *D* = durchschnittliche Anzahl doppelter Werte pro Schlüsselwert
   - *K* = *durchschnittliche Schlüsselgröße*

   Beachten Sie, dass die Werte für *minimale Schlüsselgröße* und *durchschnittliche Schlüsselgröße* ein Byte extra für jeden Teil des Schlüssels vorsehen müssen, der einen Nullwert enthalten kann, und zusätzliche zwei Byte für die Länge jedes Teils des Schlüssels mit variabler Länge.

   Wenn INCLUDE-Spalten vorhanden sind, müssen sie in den Werten für *minimale Schlüsselgröße* und *durchschnittliche Schlüsselgröße* berücksichtigt werden.

Die *durchschnittliche Indexschlüsselgröße* ist die Summe der Schlüsselkomponenten, die den Index bilden:

Fester Aufwand + variabler Aufwand + Bytezahl des SQL-Datentyps

Dabei gilt Folgendes:

- Der *feste Aufwand* beträgt 13 Byte.
- Der *variable Aufwand* ist die minimale Tiefe des indexierten Knotens plus 4 Byte.
- Der Wert für *Bytezahl des SQL-Datentyps* (sql-data-type) folgt den gleichen Regeln wie SQL.

Der Wert `0,9` kann durch einen beliebigen, mit (100 - pctfree)/100 berechneten Wert ersetzt werden, wenn während der Indexerstellung ein anderer Prozentwert für freien Speicherbereich (pctfree) angegeben wird als der Standardwert 10.

2. Die Durchschnittszahl von Schlüsseln pro Blattseite kann mit folgender Formel etwas genauer abgeschätzt werden:

   Blattseitenzahl = x / (Durchschnittszahl von Schlüsseln pro Blattseite))

   Dabei ist *x* die Gesamtzahl der Zeilen in der Tabelle bzw. der Partition.

   Für den Index für eine XML-Spalte, ist *x* die Gesamtzahl der indexierten Knoten in der Spalte.

   Einen Schätzwert für die Originalgröße eines Index können Sie wie folgt berechnen:

   $(L + 2L/(\textit{Durchschnittszahl von Schlüsseln pro Blattseite})) \times \textit{Seitengröße}$

   Für DMS-Tabellenbereiche addieren Sie die Größen aller Indizes einer Tabelle und runden auf ein Vielfaches des Werts für EXTENTSIZE für den Tabellenbereich auf, in dem sich der Index befindet.

   Sie sollten weiteren Speicherbereich für das Anwachsen des Index durch INSERT/UPDATE-Vorgänge bereitstellen, die zur Teilung von Seiten führen können.

   Durch die folgende Berechnung erhalten Sie einen genaueren Schätzwert für die Originalgröße des Index sowie einen Schätzwert für die Anzahl der Indexstufen. (Dies ist möglicherweise von besonderem Interesse, wenn in der Indexdefinition INCLUDE-Spalten verwendet werden.) Die Durchschnittszahl von Schlüsseln pro Nichtblattseite kann mit folgender Formel grob abgeschätzt werden:

   $((0{,}9 \times (U - (M \times 2))) \times (D + 1)) \div (K + 13 + (9 * D))$

   Dabei gilt Folgendes:

   - *U*, der verwendbare Speicherbereich auf einer Seite, entspricht ungefähr der Seitengröße minus 100. Bei einer Seitengröße von 4096 ist U gleich 3996.
   - *D* ist die durchschnittliche Anzahl doppelter Werte pro Schlüsselwert auf Nichtblattseiten (dieser Wert ist deutlich kleiner als für Blattseiten. Sie können ihn zur Vereinfachung der Berechnung auf 0 setzen).
   - *M = U ÷ (9 + minimale Schlüsselgröße* für Nichtblattseiten)
   - *K = durchschnittliche Schlüsselgröße* für Nichtblattseiten

   Die *minimale Schlüsselgröße* und die *durchschnittliche Schlüsselgröße* für Nichtblattseiten stimmen mit den Werten für Blattseiten überein, sofern keine INCLUDE-Spalten vorkommen. INCLUDE-Spalten werden auf Nichtblattseiten nicht gespeichert.

Sie sollten 0,9 nur durch (100 - *pctfree*)÷100 ersetzen, wenn dieser Wert größer als 0,9 ist, weil auf Nichtblattseiten bei der Indexerstellung maximal 10 % Speicherbereich frei bleiben.

Die Zahl der Nichtblattseiten kann mit folgender Formel abgeschätzt werden:

```
if L > 1 then {P++; Z++}
While (Y > 1)
{
   P = P + Y
   Y = Y / N
  Z++
}
```

Dabei gilt Folgendes:

- *P* ist die Anzahl von Seiten (anfangs 0).
- *L* ist die Anzahl der Blattseiten.
- *N* ist die Anzahl von Schlüsseln pro Nichtblattseite.
- $Y = L \div N$
- *Z* ist die Anzahl der Stufen in der Indexbaumstruktur (anfangs 1).

Anmerkung: Die obige Berechnung gilt für einen einzelnen, nicht partitionierten Index oder für eine einzelne Indexpartition bei partitionierten Indizes.

Als Gesamtzahl der Seiten ergibt sich:

$$T = (L + P + 2) \times 1{,}0002$$

Die zusätzlichen 0,02 % (1,0002) sind für Systemaufwand (einschließlich Speicherzuordnungsseiten).

Der für die Indexerstellung erforderliche Speicherbereich lässt sich folgendermaßen abschätzen:

$$T \times \text{Seitengröße}$$

Indexkomprimierung

Indizes, einschließlich Indizes für deklarierte oder erstellte temporäre Tabellen, können komprimiert werden, um Speicherkosten zu senken. Dies ist insbesondere für große OLTP- und Data-Warehouse-Umgebungen nützlich.

Standardmäßig ist die Indexkomprimierung für komprimierte Tabelle aktiviert und für nicht komprimierte Tabellen inaktiviert. Sie können dieses Standardverhalten ändern, indem Sie die Option **COMPRESS YES** der Anweisung CREATE INDEX verwenden. Wenn Sie mit vorhandenen Indizes arbeiten, können Sie die Indexkomprimierung mithilfe der Anweisung ALTER INDEX aktivieren oder inaktivieren. In diesem Fall müssen Sie anschließend eine Indexreorganisation ausführen, um den Index erneut zu erstellen.

Einschränkung: Die Indexkomprimierung wird für die folgenden Typen von Indizes nicht unterstützt:

- Blockindizes
- XML-Pfadindizes

Darüber hinaus gilt Folgendes:

- Indexspezifikationen können nicht komprimiert werden.
- Komprimierungsattribute für Indizes für temporäre Tabellen können nicht mit dem Befehl ALTER INDEX geändert werden.

Wenn die Indexkomprimierung aktiviert wird, wird das Format von Indexseiten auf der Platte und im Hauptspeicher entsprechend den Komprimierungsalgorithmen geändert, die vom Datenbankmanager zur Minimierung des Speicherplatzes ausgewählt werden. Der Grad der Komprimierung, der erreicht werden kann, wird nach Typ des erstellten Index sowie abhängig von den im Index enthaltenen Daten variieren. Der Datenbankmanager kann zum Beispiel einen Index mit einer hohen Anzahl von doppelten Schlüsseln komprimieren, indem er ein abgekürztes Format der Satz-ID (RID) für die doppelten Schlüssel speichert. In einem Index, der durch einen hohen Grad an Gemeinsamkeit in den Präfixen der Indexschlüssel gekennzeichnet ist, kann der Datenbankmanager die Komprimierung auf der Basis der Ähnlichkeiten in Präfixen von Indexschlüsseln anwenden.

Die Komprimierung kann Einschränkungen unterliegen und mit gewissen Kosten verbunden sein. Wenn die Indizes keine gemeinsamen Indexspaltenwerte oder gemeinsame Teile von Präfixen enthalten, können die Vorteile der Indexkomprimierung in Hinblick auf verringerten Speicherbedarf vernachlässigbar sein. Und obwohl ein eindeutiger Index für eine Zeitmarkenspalte möglicherweise ein sehr hohes Komprimierungspotenzial aufgrund von gemeinsamen Werten für Jahr, Monat, Tag, Stunde, Minute oder sogar Sekunden auf derselben Blattseite hat, könnte die Untersuchung, ob gemeinsame Präfixe vorhanden sind, zu Leistungseinbußen führen.

Wenn Sie glauben, dass die Komprimierung in Ihrer speziellen Situation keinen Vorteil bietet, können Sie die Indizes entweder ohne Komprimierung erneut erstellen oder die Indizes ändern und anschließend eine Indexreorganisation ausführen, um die Indexkomprimierung zu inaktivieren.

Wenn Sie die Indexkomprimierung in Betracht ziehen, müssen Sie einige Faktoren berücksichtigen:

- Wenn Sie die Zeilenkomprimierung mit der Option **COMPRESS YES** im Befehl CREATE TABLE oder ALTER TABLE aktivieren, wird die Komprimierung standardmäßig für alle Indizes, für die sie unterstützt wird und die nach diesem Zeitpunkt für die Betreffende Tabelle erstellt werden, aktiviert, sofern sie nicht explizit durch die Befehle CREATE INDEX oder ALTER INDEX inaktiviert wird. Analog wird die Komprimierung, wenn Sie sie mithilfe des Befehls CREATE TABLE bzw. ALTER TABLE inaktiviert haben, für alle Indizes inaktiviert, die nach diesem Zeitpunkt für die Tabelle erstellt werden, sofern sie nicht explizit durch die Befehle CREATE INDEX oder ALTER INDEX aktiviert wird.
- Wenn Sie die Indexkomprimierung mithilfe des Befehls ALTER INDEX aktivieren, erfolgt die Komprimierung erst, nachdem eine Indexreorganisation ausgeführt wurde. Umgekehrt bleibt der Index, wenn Sie die Komprimierung inaktivieren, komprimiert, bis Sie eine Indexreorganisation ausführen.
- Bei einer Datenbankmigration wird die Komprimierung für alle Indizes, die möglicherweise migriert wurden, nicht aktiviert. Wenn die Komprimierung verwendet werden soll, müssen Sie den Befehl ALTER INDEX und anschließend eine Indexreorganisation ausführen.
- Die CPU-Belastung kann sich infolge der Verarbeitung, die für die Indexkomprimierung bzw. Indexdekomprimierung erforderlich ist, leicht erhöhen. Falls dies nicht akzeptabel ist, können Sie die Indexkomprimierung für neue oder vorhandene Indizes inaktivieren.

Beispiele

Beispiel 1: Überprüfen, ob ein Index komprimiert ist.

Durch die beiden folgenden Anweisungen werden eine neue Tabelle T1, für die die Zeilenkomprimierung aktiviert ist, und ein Index I1 für T1 erstellt.

```
CREATE TABLE T1 (C1 INT, C2 INT, C3 INT) COMPRESS YES
CREATE INDEX I1 ON T1(C1)
```

Standardmäßig werden Indizes für T1 komprimiert. Das *Komprimierungsattribut* für den Index I1, das zeigt, ob die Komprimierung aktiviert ist, kann mithilfe der Katalogtabelle oder der entsprechenden ADMIN-Tabellenfunktion überprüft werden:

```
SELECT COMPRESSION FROM SYSCAT.INDEXES WHERE TABNAME='T1'

COMPRESSION
-----------
Y

   1 Satz/Sätze ausgewählt.
```

Beispiel 2: Ermitteln, ob komprimierte Indizes eine Reorganisation erfordern.

Zur Überprüfung, ob komprimierte Indizes eine Reorganisation erfordern, verwenden Sie den Befehl **REORGCHK**. Tabelle 78 auf Seite 499 zeigt, wie der Befehl für eine Tabelle mit dem Namen T1 ausgeführt wird:

Tabelle 78. Ausgabe des Befehls **REORGCHK**

```
REORGCHK ON TABLE SCHEMA1.T1

Ausführung von RUNSTATS ....

Tabellenstatistik:

F1: 100 * OVERFLOW / CARD < 5
F2: 100 * (Effektive Speicherauslaustung von Datenseiten) > 70
F3: 100 * (Erforderliche Seiten / Seiten insgesamt) > 80

SCHEMA.NAME              CARD    OV    NP    FP ACTBLK   TSIZE  F1  F2  F3 REORG
-------------------------------------------------------------------------------
Tabelle: SCHEMA1.T1

                          879     0    14    14     -   51861   0 100 100 ---
-------------------------------------------------------------------------------

Indexstatistik:

F4: CLUSTERRATIO oder normalisierter CLUSTERFACTOR > 80
F5: 100 * (Für Blattseiten verwendeter Speicherbereich / Verfügbarer Speicherbereich für nicht leere Blattseiten) > MIN(50, (100 - PCTFREE))
F6: (100 - PCTFREE) * (Verfügbarer Speicherbereich in einem Index mit einer Ebene weniger / Erforderlicher Speicherbereich für alle Schlüssel) < 100
F7: 100 * (Anzahl pseudogelöschter Satz-IDs / Gesamtzahl Satz-IDs) < 20
F8: 100 * (Anzahl pseudoleerer Blattseiten / Gesamtzahl Blattseiten) < 20

SCHEMA.NAME              INDCARD  LEAF ELEAF LVLS  NDEL   KEYS LEAF_RECSIZE NLEAF_RECSIZE LEAF_PAGE_OVERHEAD NLEAF_PAGE_OVERHEAD  PCT_PAGES_SAVED  F4  F5  F6  F7  F8 REORG
-------------------------------------------------------------------------------------------------------------------------------------------------------
Tabelle: SCHEMA1.T1
Index: SCHEMA1.I1
                          879    15     0    2     0    682           20                 20                 596                 596          28  56  31   -   0   0 -----
-------------------------------------------------------------------------------------------------------------------------------------------------------
```

Beispiel 3: Ermitteln der potenziellen Speichereinsparungen der Indexkomprimierung.

Ein Beispiel dazu, wie Sie die potenziellen Einsparungen der Indexkomprimierung berechnen können, finden Sie in der Dokumentation zur Tabellenfunktion ADMIN_GET_INDEX_COMPRESS_INFO.

Erstellen von Indizes

Indizes können aus verschiedenen Gründen erstellt werden, unter anderen, um Abfragen effizienter auszuführen, die Zeilen einer Tabelle aufsteigend oder absteigend nach den Werten einer Spalte zu sortieren oder um Integritätsbedingungen wie zum Beispiel für die Eindeutigkeit der Indexschlüssel zu erzwingen. Sie können die Anweisung CREATE INDEX, den DB2-Designadvisor oder den Designadvisorbefehl **db2advis** verwenden, um die Indizes zu erstellen.

Vorbereitende Schritte

Auf Solaris-Plattformen ist Patch 122300-11 für Solaris 9 bzw. 125100-07 für Solaris 10 erforderlich, um Indizes auf Roheinheiten (RAW) erstellen zu können. Ohne dieses Patch wird die Anweisung CREATE INDEX blockiert, wenn eine Roheinheit verwendet wird.

Informationen zu diesem Vorgang

Diese Task geht davon aus, dass Sie einen Index für eine nicht partitionierte Tabelle erstellen wollen.

Vorgehensweise

Verwenden Sie zum Erstellen eines Index über die Befehlszeile die Anweisung CREATE INDEX.
Beispiel:

```
CREATE UNIQUE INDEX EMP_IX
  ON EMPLOYEE(EMPNO)
  INCLUDE(FIRSTNAME, JOB)
```

Die Klausel INCLUDE, die nur für eindeutige Indizes verwendbar ist, gibt zusätzliche Spalten an, die an die Gruppe der Indexschlüsselspalten angehängt werden. Alle Spalten, die mit dieser Klausel in den Index eingeschlossen werden, werden nicht zur Umsetzung der Eindeutigkeit verwendet. Solche INCLUDE-Spalten können die Leistung einiger Abfragen durch die Möglichkeit eines reinen Indexzugriffs verbessern. Diese Option bietet folgende Vorteile:

- Sie hilft bei der Vermeidung ansonsten notwendiger Zugriffe auf die Datenseiten für weitere Abfragen.
- Sie hilft bei der Vermeidung redundanter Indizes.

Wenn die Anweisung SELECT EMPNO, FIRSTNAME, JOB FROM EMPLOYEE an der Tabelle, für die dieser Index definiert ist, ausgeführt wird, können alle angeforderten Daten aus dem Index abgerufen werden, ohne dass die Datenseiten gelesen werden müssen. Dies verbessert die Leistung.

Nächste Schritte

Wenn eine Zeile gelöscht oder aktualisiert wird, werden die Indexschlüssel als gelöscht markiert, jedoch erst physisch von der Seite entfernt, wenn einige Zeit nach dem Festschreiben der Löschung bzw. Aktualisierung eine Bereinigung erfolgt. Solche Schlüssel werden als pseudogelöscht bezeichnet. Eine solche Bereinigung kann zum Beispiel durch eine nachfolgende Transaktion durchgeführt werden, welche die Seite ändert, auf der der Schlüssel als gelöscht markiert ist. Die Bereinigung pseudogelöschter Schlüssel kann explizit durch Verwenden des Parameters **CLEANUP ONLY ALL** im Befehl **REORG INDEXES** ausgelöst werden.

Erstellen nicht partitionierter Indizes für partitionierte Tabellen

Wenn Sie einen *nicht partitionierten Index* für eine partitionierte Tabelle erstellen, erstellen Sie ein einzelnes Indexobjekt, das sich auf alle Zeilen in der Tabelle bezieht. Nicht partitionierte Indizes werden immer in nur einem Tabellenbereich erstellt, selbst wenn sich die Tabellendatenpartitionen über mehrere Tabellenbereiche erstrecken.

Vorbereitende Schritte

Diese Task geht von der Annahme aus, dass die partitionierte Tabelle bereits erstellt wurde.

Vorgehensweise

1. Formulieren Sie eine Anweisung CREATE INDEX für Ihre Tabelle mit der Klausel NOT PARTITIONED. Beispiel:

   ```
   CREATE INDEX indexName ON tabellenName(spalte) NOT PARTITIONED
   ```

2. Führen Sie die Anweisung CREATE INDEX über eine unterstützte DB2-Schnittstelle aus.

Beispiel

Beispiel 1: Erstellen eines nicht partitionierten Index in demselben Tabellenbereich wie die Datenpartition

Nehmen Sie, die Tabelle SALES ist wie folgt definiert:

```
CREATE TABLE sales(store_num INT,
                   sales_date DATE,
                   total_sales DECIMAL (6,2)) IN ts1
   PARTITION BY RANGE(store_num)
   (STARTING FROM (1) ENDING AT (100),
    STARTING FROM (101) ENDING AT (150),
    STARTING FROM (151) ENDING AT (200))
```

Die drei Partitionen der Tabelle SALES sind in Tabellenbereich TS1 gespeichert. Standardmäßig werden alle Indizes, die für diese Tabelle erstellt werden, auch in TS1 gespeichert, weil dies der Tabellenbereich ist, der für diese Tabelle angegeben wurde. Verwenden Sie die folgende Anweisung, um einen nicht partitionierten Index mit dem Namen STORENUM für die Spalte STORE_NUM zu erstellen:

```
CREATE INDEX StoreNum ON sales(store_num) NOT PARTITIONED
```

Beachten Sie, dass die Klausel NOT PARTITIONED erforderlich ist, da der Index ansonsten dem Standardverhalten für partitionierte Tabellen entsprechend als partitionierter Index erstellt wird.

Beispiel 2: Erstellen eines nicht partitionierten Index in einem anderen Tabellenbereich als dem Standardtabellenbereich

Für dieses Beispiel wird angenommen, dass die Tabelle PARTS wie folgt definiert ist:

```
CREATE TABLE parts(part_number INT,
                   manufacturer CHAR,
                   description CLOB,
                   price DECIMAL (4,2)) IN ts1 INDEX in ts2
   PARTITION BY RANGE (part_number)
   (STARTING FROM (1) ENDING AT (10) IN ts3,
    STARTING FROM (11) ENDING AT (20) INDEX IN ts1,
    STARTING FROM (21) ENDING AT (30) IN ts2 INDEX IN ts4);
```

Die Tabelle PARTS besteht aus drei Partitionen: die erste in Tabellenbereich TS3, die zweite in TS1 und die dritte in TS2. Wenn Sie die folgende Anweisung ausführen, wird ein nicht partitionierter Index erstellt, der die Zeilen in absteigender Reihenfolge nach Namen des Herstellers ('manufacturer') sortiert:

```
CREATE INDEX manufct on parts(manufacturer DESC) NOT PARTITIONED IN TS3;
```

Dieser Index wird in Tabellenbereich TS3 erstellt. Die Klausel INDEX IN der Anweisung CREATE TABLE wird durch die Klausel IN *tabellenbereich* der Anweisung CREATE INDEX überschrieben. Da die Tabelle PARTS partitioniert ist, müssen Sie die Klausel NOT PARTITIONED in der Anweisung CREATE INDEX angeben, um einen nicht partitionierten Index zu erstellen.

Erstellen partitionierter Indizes

Wenn Sie einen *partitionierten Index* für eine partitionierte Tabelle erstellen, wird jede Datenpartition in einer eigenen Indexpartition indexiert. Standardmäßig wird die Indexpartition in demselben Tabellenbereich wie die durch sie indexierte Datenpartition gespeichert. Die Daten in den Indizes werden auf der Basis des Verteilungsschlüssels der Tabelle verteilt.

Vorbereitende Schritte

Diese Task geht von der Annahme aus, dass die partitionierte Tabelle bereits erstellt wurde.

Informationen zu diesem Vorgang

Einschränkungen

Es gibt einige Typen von Indizes, die nicht partitioniert werden können:

- Indizes zu nicht partitionierten Daten
- Indizes zu räumlichen Daten
- XML-Spaltenpfadindizes (vom System generiert)

Sie müssen diese Indizes immer als nicht partitionierte Indizes erstellen. Darüber hinaus muss der Indexschlüssel für partitionierte eindeutige Indizes alle Spalten aus dem Tabellenpartitionierungsschlüssel enthalten, und zwar unabhängig davon, ob sie von einem Benutzer oder vom System generiert wurden. Das Letztere wäre der Fall für Indizes, die vom System generiert werden, um Integritätsbedingungen für Eindeutigkeit oder Primärschlüssel für Daten umzusetzen.

Außerdem wird die Klausel IN der Anweisung CREATE INDEX für die Erstellung partitionierter Indizes nicht unterstützt. Standardmäßig werden Indexpartitionen in demselben Tabellenbereich wie die durch sie indexierten Datenpartitionen erstellt. Zur Angabe eines alternativen Tabellenbereichs zum Speichern der Indexpartition müssen Sie die Klausel INDEX IN der Anweisung CREATE TABLE auf Partitionsebene verwenden, um einen Tabellenbereich für Indizes auf der Basis der einzelnen Partitionen anzugeben. Wenn Sie diese Klausel nicht angeben, werden die Indexpartitionen in demselben Tabellenbereich wie die Datenpartitionen, die sie indexieren, gespeichert.

Vorgehensweise

1. Formulieren Sie eine Anweisung CREATE INDEX für Ihre Tabelle mit der Klausel PARTITIONED.
2. Führen Sie die Anweisung CREATE INDEX über eine unterstützte DB2-Schnittstelle aus.

Beispiel

Anmerkung: Diese Beispiele dienen nur zu Veranschaulichungszwecken und geben keine Hinweise auf bewährte Verfahren zur Erstellung von partitionierten Tabellen oder Indizes.
Beispiel 1: Erstellen eines partitionierten Index in denselben Tabellenbereichen wie die jeweilige Datenpartition

In diesem Beispiel wird angenommen, dass die Tabelle SALES wie folgt definiert wurde:

```
CREATE TABLE sales(store_num INT,
                   sales_date DATE,
                   total_sales DECIMAL (6,2)) IN ts1
 PARTITION BY RANGE(store_num)
   (STARTING FROM (1) ENDING AT (100),
    STARTING FROM (101) ENDING AT (150),
    STARTING FROM (151) ENDING AT (200))
```

In diesem Fall werden die drei Partitionen der Tabelle SALES in Tabellenbereich `ts1` gespeichert. Alle partitionierten Indizes, die für diese Tabelle erstellt werden, werden ebenfalls in `ts1` gespeichert, weil dies der Tabellenbereich ist, in dem jede Partition für diese Tabelle gespeichert wird. Führen Sie die folgende Anweisung aus, um einen partitionierten Index für die Spalte 'store_num' (Lagernummer) zu erstellen:

```
CREATE INDEX StoreNum ON sales(store_num) PARTITIONED
```

Beispiel 2: Auswählen einer alternativen Position für alle Indexpartitionen

In diesem Beispiel wird angenommen, dass die Tabelle EMPLOYEE wie folgt definiert wurde:

```
CREATE TABLE employee(employee_number INT, employee_name CHAR,
             job_code INT, city CHAR, salary DECIMAL (6,2))
             IN ts1 INDEX IN ts2
   PARTITION BY RANGE (job_code)
   (STARTING FROM (1) ENDING AT (10) INDEX IN ts2,
    STARTING FROM (11) ENDING AT (20) INDEX IN ts2,
    STARTING FROM (21) ENDING AT (30) INDEX IN ts2)
```

Führen Sie die folgende Anweisung aus, um einen partitionierten Index für die Spalte 'job_code' (Tätigkeitscode) zu erstellen:

```
CREATE INDEX JobCode ON employee(job_code) PARTITIONED
```

In diesem Beispiel werden die Partitionen der Tabelle EMPLOYEE in Tabellenbereich `ts1` gespeichert. Sämtliche Indexpartitionen werden jedoch in Tabellenbereich `ts2` gespeichert.

Beispiel 3: In verschiedenen Partitionen erstellte Indizes

Für dieses Beispiel wird angenommen, dass die Tabelle PARTS wie folgt definiert wurde:

```
CREATE TABLE parts(part_number INT, manufacturer CHAR, description CLOB,
price DECIMAL (4,2)) IN ts1 INDEX in ts2
   PARTITION BY RANGE (part_number)
   (STARTING FROM (1) ENDING AT (10) IN ts3,
    STARTING FROM (11) ENDING AT (20) INDEX IN ts1,
    STARTING FROM (21) ENDING AT (30) IN ts2 INDEX IN ts4);
```

In diesem Fall besteht die Tabelle PARTS aus drei Partitionen: die erste befindet sich in Tabellenbereich ts3, die zweite in ts1 und die dritte in ts2. Nun werden die folgenden Anweisungen ausgeführt:

```
CREATE INDEX partNoasc ON parts(part_number ASC)  PARTITIONED
CREATE INDEX manufct on parts(manufacturer DESC) NOT PARTITIONED IN TS3;
```

Durch diese Anweisungen werden zwei Indizes erstellt. Der erste Index ist ein partitionierter Index, der zum Sortieren der Zeilen in aufsteigender Reihenfolge nach Teilenummer ('part_number') dient. Die erste Indexpartition wird in Tabellenbereich ts3 erstellt, die zweite in ts1 und die dritte in ts4. Der zweite Index ist ein nicht partitionierter Index, der die Zeilen in absteigender Reihenfolge nach dem Namen des Herstellers ('manufacturer') sortiert. Dieser Index wird in Tabellenbereich ts3 erstellt. Beachten Sie, dass die Klausel IN in Anweisungen CREATE INDEX für nicht partitionierte Indizes zulässig ist. In diesem Fall muss außerdem, da die Tabelle PARTS partitioniert ist, die Klausel NOT PARTITIONED in der Anweisung CREATE INDEX angegeben werden, um einen nicht partitionierten Index zu erstellen.

Ändern von Indizes

Wenn Sie einen Index ändern möchten, können Sie dies nicht wie bei der Aktivierung oder Inaktivierung der Indexkomprimierung mit der Anweisung ALTER INDEX tun, sondern müssen den Index löschen und anschließend einen neuen Index erstellen.

Beispiel

Es ist zum Beispiel nicht möglich, der Liste der Schlüsselspalten eine Spalte hinzuzufügen, ohne die vorherige Definition zu löschen und einen neuen Index zu erstellen. Sie können jedoch einen Kommentar mithilfe der Anweisung COMMENT hinzufügen, um den Zweck des Index zu beschreiben.

Umbenennen von Indizes

Mithilfe der Anweisung RENAME können Sie einen vorhandenen Index umbenennen.

Informationen zu diesem Vorgang

Beim Umbenennen eines Index darf der Quellenindex kein systemgenerierter Index sein.

Vorgehensweise

Zum Umbenennen eines vorhandenen Index geben Sie die folgende Anweisung in die Befehlszeile ein:

```
RENAME INDEX quellenindexname TO zielindexname
```

Die Angabe *quellenindexname* ist der Name des vorhandenen Index, der umbenannt werden soll. Der Name, einschließlich des Schemanamens, muss einen Index angeben, der in der Datenbank vorhanden ist. Dabei darf es sich nicht um den Namen eines Index für eine deklarierte temporäre Tabelle oder eine erstellte temporäre Tabelle handeln. Der Schemaname darf nicht SYSIBM, SYSCAT, SYSFUN oder SYSSTAT sein.
Die Angabe *zielindexname* enthält den neuen Namen für den Index ohne Schemana-

men. Der Schemaname des Quellenobjekts wird zur Qualifikation des neuen Namens für das Objekt verwendet. Der qualifizierte Name darf keinen Index bezeichnen, der in der Datenbank vorhanden ist.

Ergebnisse

Wenn die Anweisung RENAME erfolgreich ausgeführt wird, werden die Systemkatalogtabellen mit dem neuen Indexnamen aktualisiert.

Erneutes Erstellen von Indizes

Bestimmte Datenbankoperationen, wie zum Beispiel eine aktualisierende Recovery (Rollforward), die eine Indexerstellungsoperation (CREATE INDEX) umfasst, die nicht vollständig protokolliert wurde, können dazu führen, dass ein Indexobjekt ungültig wird, weil der Index während der aktualisierenden Recovery nicht erstellt wird. Das Indexobjekt kann durch erneutes Erstellen des enthaltenen Index wiederhergestellt werden.

Informationen zu diesem Vorgang

Wenn der Datenbankmanager erkennt, dass ein Index nicht mehr gültig ist, versucht er automatisch, diesen Index erneut zu erstellen. Wann die erneute Erstellung stattfindet, wird durch den Parameter **indexrec** in der Konfigurationsdatei der Datenbank oder des Datenbankmanagers gesteuert. Für diesen Parameter sind fünf Einstellungen möglich:

- SYSTEM
- RESTART
- RESTART_NO_REDO
- ACCESS
- ACCESS_NO_REDO

Die Werte RESTART_NO_REDO und ACCESS_NO_REDO haben eine ähnliche Bedeutung wie RESTART und ACCESS.

Die NO_REDO-Optionen legen fest, dass, auch wenn der Index während der ursprünglichen Operation (z. B. CREATE INDEX) vollständig protokolliert wurde, der Index bei der aktualisierende Recovery nicht erneut erstellt wird. Stattdessen wird er entweder beim Neustart oder beim ersten Zugriff erstellt. Weitere Informationen finden Sie in der Beschreibung des Parameters **indexrec**.

Wenn der Zeitpunkt des Datenbankneustarts kein Problem darstellt, ist es besser, ungültige Indizes im Rahmen des Prozesses zur Wiederherstellung der Datenbank in einem konsistenten Status erneut zu erstellen. Bei dieser Vorgehensweise dauert das erneute Starten einer Datenbank aufgrund des erneuten Erstellungsprozesses für den Index länger. Jedoch wird der normale Verarbeitungsbetrieb nicht mehr gestört, wenn sich die Datenbank wieder in einem konsistenten Status befindet.

Wenn Indizes andererseits beim nächsten Zugriff erneut erstellt werden, ist die Zeit, die für den Neustart einer Datenbank benötigt wird, zwar kürzer, jedoch kann es anschließend infolge eines Index, der erneut erstellt wird, zu einer unerwarteten Verschlechterung der Antwortzeit kommen. Zum Beispiel müssen Benutzer, die auf eine Tabelle zugreifen, die einen ungültigen Index hat, warten, bis der Index erneut erstellt wurde. Darüber hinaus ist es möglich, dass unerwartete Sperren aktiviert und bis lange nach der Neuerstellung eines ungültigen Index beibehalten werden, insbesondere wenn die Transaktion, die die Neuerstellung des Index verursacht hat, nie beendet wird (d. h. die durch sie vorgenommenen Änderungen festschreibt (COMMIT) oder rückgängig macht (ROLLBACK)).

Löschen von Indizes

Verwenden Sie zum Löschen eines Index die Anweisung DROP.

Informationen zu diesem Vorgang

Abgesehen vom Attribut COMPRESSION eines Index können Sie keine Klausel einer Indexdefinition ändern. Sie müssen den Index löschen und einen neuen Index erstellen. Das Löschen eines Index führt nicht dazu, dass andere Objekte gelöscht werden. Allerdings werden möglicherweise einige Pakete inaktiviert bzw. ungültig gemacht.

Einschränkungen

Ein Index für den Primärschlüssel oder eindeutigen Schlüssel kann nicht explizit gelöscht werden. Zum Löschen eines solchen Index gibt es folgende Methoden:

- Wenn der Primärindex bzw. die eindeutige Integritätsbedingung automatisch für den Primärschlüssel oder eindeutigen Schlüssel erstellt wurde, wird der Index durch Löschen des Primärschlüssels oder eindeutigen Schlüssels gelöscht. Das Löschen des Schlüssels erfolgt mithilfe der Anweisung ALTER TABLE.
- Wenn der Primärindex oder die eindeutige Integritätsbedingung benutzerdefiniert ist, muss der Primärschlüssel oder eindeutige Schlüssel zuerst mit der Anweisung ALTER TABLE gelöscht werden. Nach dem Löschen des Primärschlüssels oder des eindeutigen Schlüssels wird der Index nicht länger als Primärindex oder eindeutiger Index betrachtet und kann daher explizit gelöscht werden.

Vorgehensweise

Geben Sie zum Löschen eines Index über die Befehlszeile die folgende Anweisung ein:

```
DROP INDEX indexname
```

Ergebnisse

Pakete und zwischengespeicherte dynamische SQL- und XQuery-Anweisungen, die von den gelöschten Indizes abhängig sind, werden als ungültig markiert. Das Anwendungsprogramm ist von Änderungen durch das Hinzufügen oder Löschen von Indizes nicht betroffen.

Kapitel 15. Trigger

Ein *Trigger* definiert eine Reihe von Aktionen, die in Reaktion auf eine Einfüge-, Aktualisierungs- oder Löschoperation für eine angegebene Tabelle ausgeführt werden. Wenn eine solche SQL-Operation ausgeführt wird, wird der Trigger als *aktiviert* bezeichnet. Trigger sind optional und werden mithilfe der Anweisung CREATE TRIGGER definiert.

Trigger können zusammen mit referenziellen Integritätsbedingungen und Prüfungen auf Integritätsbedingungen in Tabellen zur Implementierung von Datenintegritätsregeln eingesetzt werden. Darüber hinaus können Trigger auch genutzt werden, um Aktualisierungen an anderen Tabellen zu bewirken, automatisch Werte für eingefügte oder aktualisierte Zeilen zu generieren bzw. umzuwandeln oder Funktion für solche Operationen wie das Absetzen von Alerts aufzurufen.

Trigger bilden einen nützlichen Mechanismus zur Definition und Implementierung von *Übergangsregeln* für den Geschäftsbetrieb, bei denen es sich um Regeln handelt, die für verschiedene Status der Daten gelten (z. B. ein Gehalt, dass nicht um mehr als zehn Prozent angehoben werden kann).

Mithilfe von Triggern lässt sich die Logik, durch die Geschäftsregeln implementiert werden, außerhalb der Datenbank verwalten. Das heißt, dass nicht die Anwendungen für die Implementierung dieser Regeln zuständig sind. Eine zentralisierte Logik, die für alle Tabellen implementiert wird, ermöglicht eine einfachere Pflege, da keine Änderungen an Anwendungsprogrammen erforderlich werden, wenn sich die Logik ändert.

Zur Erstellung eines Triggers sind folgende Angaben erforderlich:

- Die *Subjekttabelle* gibt die Tabelle an, für die der Trigger definiert wird.
- Das *Trigger-Ereignis* definiert eine bestimmte SQL-Operation, die die Subjekttabelle modifiziert. Bei dem Ereignis kann es sich um eine Einfüge-, Aktualisierungs- oder Löschoperation handeln.
- Die *Aktivierungszeit des Triggers* gibt an, ob der Trigger vor oder nach dem Eintreten des Triggerereignisses zu aktivieren ist.

Die Anweisung, welche die Aktivierung eines Triggers bewirkt, enthält eine *Gruppe der betroffenen Zeilen*. Dies sind die Zeilen der Subjekttabelle, die eingefügt, aktualisiert oder gelöscht werden. Die *Triggergranularität* gibt an, ob die Aktionen des Triggers einmal für die Anweisung oder einmal für jede der betroffenen Zeilen ausgeführt werden.

Die *ausgelöste Aktion* besteht aus einer optionalen Suchbedingung und einer Gruppe von Anweisungen, die ausgeführt werden, wenn der Trigger aktiviert wird. Die Anweisungen werden nur ausgeführt, wenn die Suchbedingung ein wahres Ergebnis liefert. Wenn die Aktivierungszeit des Triggers vor dem Trigger-Ereignis liegt, können die ausgelösten Aktionen Anweisungen umfassen, mit denen Auswahlen getroffen, Übergangsvariablen festgelegt oder SQL-Status signalisiert werden können. Liegt die Aktivierungszeit des Triggers nach dem Trigger-Ereignis, können die ausgelösten Aktionen Anweisungen umfassen, die auswählen (SELECT), aktualisieren (UPDATE), löschen (DELETE) oder SQL-Status signalisieren.

Die ausgelöste Aktion kann sich mithilfe von *Übergangsvariablen* auf die Werte in der Gruppe der betroffenen Zeilen beziehen. Übergangsvariablen verwenden die Namen der Spalten in der Subjekttabelle, die durch einen angegebenen Namen qualifiziert werden, der zu erkennen gibt, ob es sich um einen Verweis auf den alten Wert (vor der Aktualisierung) oder den neuen Wert (nach der Aktualisierung) handelt. Der neue Wert kann mithilfe der Anweisung 'SET Variable' in BEFORE-, INSERT und UPDATE-Triggern ebenfalls geändert werden.

Eine weitere Methode zur Bezugnahme auf die Werte in der Gruppe der betroffenen Zeilen besteht in der Verwendung von *Übergangstabellen*. Übergangstabellen können ebenfalls die Namen von Spalten in der Subjekttabelle verwenden, stellen jedoch einen Namen bereit, über den die gesamte Gruppe der betroffenen Zeilen wie eine Tabelle behandelt werden kann. Übergangstabellen können nur in AFTER-Triggern (d. h., nicht mit BEFORE- und INSTEAD OF-Triggern) verwendet werden, wobei für alte und neue Werte getrennte Übergangstabellen definiert werden können.

Es können mehrere Trigger angegeben werden, um Tabellen, Ereignisse (INSERT, UPDATE, DELETE) oder Aktivierungszeiten (BEFORE, AFTER, INSTEAD OF) zu kombinieren. Wenn mehr als ein Trigger für eine bestimmte Tabelle, ein bestimmtes Ereignis und eine bestimmte Aktivierungszeit vorhanden ist, entspricht die Reihenfolge, in der die Trigger aktiviert werden, der Reihenfolge, in der sie erstellt werden. Das heißt, der zuletzt erstellte Trigger wird auch zuletzt aktiviert.

Die Aktivierung eines Triggers kann unter Umständen zu einem *Hintereinanderschalten von Triggern* führen. Diese Bezeichnung bezieht sich auf das Ergebnis der Aktivierung eines Triggers, der Anweisungen ausführt, die wiederum die Aktivierung anderer Trigger oder sogar die erneute Aktivierung desselben Triggers bewirken. Die ausgelösten Aktionen können außerdem Aktualisierungen bewirken, die sich aus der Anwendung von Regeln der referenziellen Integrität für Löschungen ergeben, die wiederum die Aktivierung weiterer Trigger nach sich ziehen können. Durch die Hintereinanderschaltung von Triggern kann eine Kette von Triggern und Löschregeln der referenziellen Integrität aktiviert werden, die einen erheblichen Umfang an Änderungen an der Datenbank als Folge einer einzigen INSERT-, UPDATE- oder DELETE-Anweisung bewirken.

Wenn mehrere Trigger Einfüge-, Aktualisierungs- oder Löschaktionen für dasselbe Objekt enthalten, werden zur Lösung von Zugriffskonflikten Konfliktlösungsmechanismen wie z. B. temporäre Tabellen verwendet. Dies kann sich merklich auf die Leistung auswirken, insbesondere in Umgebungen mit partitionierten Datenbanken.

Typen von Triggern

Ein *Trigger* definiert eine Reihe von Aktionen, die in Reaktion auf eine Einfüge-, Aktualisierungs- oder Löschoperation für eine angegebene Tabelle ausgeführt werden. Wenn eine solche SQL-Operation ausgeführt wird, wird der Trigger als *aktiviert* bezeichnet. Trigger sind optional und werden mithilfe der Anweisung CREATE TRIGGER definiert.

Trigger können zusammen mit referenziellen Integritätsbedingungen und Prüfungen auf Integritätsbedingungen in Tabellen zur Implementierung von Datenintegritätsregeln eingesetzt werden. Darüber hinaus können Trigger auch genutzt werden, um Aktualisierungen an anderen Tabellen zu bewirken, automatisch Werte für eingefügte oder aktualisierte Zeilen zu generieren bzw. umzuwandeln oder Funktion für solche Operationen wie das Absetzen von Alerts aufzurufen.

Folgende Typen von Triggern werden unterstützt:

BEFORE-Trigger
Diese werden vor einer Aktualisierungs- oder Einfügeoperation ausgeführt. Werte, die aktualisiert oder eingefügt wurden, können geändert werden, bevor die Datenbank tatsächlich geändert wird. Sie können Trigger, die vor einer Aktualisierungs- oder Einfügeoperation ausgeführt werden, auf verschiedene Weise verwenden:

- Zum Prüfen oder Ändern von Werten, bevor sie tatsächlich aktualisiert oder in die Datenbank eingefügt werden. Diese Art der Verwendung ist nützlich, wenn Sie Daten, so, wie sie der Benutzer sieht, in ein internes Datenbankformat umsetzen müssen.
- Zum Ausführen von anderen Operationen als Datenbankoperationen, die in benutzerdefinierten Funktionen codiert sind.

BEFORE DELETE-Trigger
Diese werden vor einer Löschoperation ausgeführt. Sie dienen zum Überprüfen von Werten (und, falls erforderlich, zum Ausgeben von Fehlern).

AFTER-Trigger
Diese werden nach einer Aktualisierungs-, Einfüge- oder Löschoperation ausgeführt. Sie können Trigger verwenden, die nach einer Aktualisierungs- oder Einfügeoperation ausgeführt werden, auf verschiedene Weise verwenden:

- Zum Aktualisieren von Daten in anderen Tabellen. Diese Funktion ist für die Pflege von Beziehungen zwischen Daten oder für die Aufbewahrung von Prüfprotokollinformationen nützlich.
- Zum Abgleichen gegen andere Daten in der Tabelle oder in anderen Tabellen. Diese Funktion ist für die Sicherstellung der Datenintegrität nützlich, wenn referenzielle Integritätsbedingungen nicht geeignet sind oder wenn Prüfungen auf Integritätsbedingung in Tabellen nur auf die Prüfung der aktuellen Tabelle beschränkt werden.
- Zum Ausführen von anderen Operationen als Datenbankoperationen, die in benutzerdefinierten Funktionen codiert sind. Diese Funktion ist nützlich, wenn Alerts abgesetzt werden; sie ist auch für die Aktualisierung von Informationen außerhalb der Datenbank hilfreich.

INSTEAD OF-Trigger
Zum Beschreiben der Vorgehensweise beim Durchführen von Einfüge-, Aktualisierung- und Löschoperationen für Sichten, die zu komplex sind, als dass sie diese Operationen nativ unterstützen könnten. Mit diesen Triggern können Anwendungen eine Sicht als alleinige Schnittstelle für alle SQL-Operationen (INSERT, DELETE, UPDATE und SELECT) verwenden.

BEFORE-Trigger

Durch die Verwendung von Triggern, die vor einer Aktualisierungs- oder Einfügeoperation ausgeführt werden, können Werte, die aktualisiert oder eingefügt wurden, geändert werden, bevor die Datenbank tatsächlich geändert wird. Falls dies gewünscht wird, können sie zur Umsetzung von Eingaben aus dem Anwendungsformat (Benutzersicht der Daten) in ein internes Datenbankformat verwendet werden.

Diese BEFORE-Trigger können auch zur Aktivierung anderer Operationen als Datenbankoperationen durch benutzerdefinierte Funktionen verwendet werden.

BEFORE-Trigger werden vor einer Löschoperation ausgeführt. Sie überprüfen die Werte und geben einen Fehler aus, falls erforderlich.

Beispiele

Im folgenden Beispiel wird ein BEFORE-Trigger zum Setzen eines komplexen Standardwerts definiert:

```
CREATE TRIGGER trigger1
  BEFORE UPDATE ON table1
  REFERENCING NEW AS N
  WHEN (N.expected_delivery_date IS NULL)
  SET N.expected_delivery_date = N.order_date + 5 days;
```

Im folgenden Beispiel wird ein BEFORE-Trigger mit einer tabellenübergreifenden Integritätsbedingung definiert, bei der es sich nicht um eine referenzielle Integritätsbedingung handelt:

```
CREATE TRIGGER trigger2
  BEFORE UPDATE ON table2
  REFERENCING NEW AS N
  WHEN (n.salary > (SELECT maxsalary FROM salaryguide WHERE rank = n.position))
  SIGNAL SQLSTATE '78000' SET MESSAGE_TEXT = 'Salary out of range');
```

AFTER-Trigger

Trigger, die nach einer Aktualisierungs-, Einfüge- oder Löschoperation ausgeführt werden, können auf verschiedene Weise verwendet werden.

- Trigger können Daten in ein und derselben Tabelle oder in anderen Tabellen aktualisieren, einfügen oder löschen. Dies ist nützlich, um Beziehungen zwischen Daten zu pflegen oder um Prüfprotokollinformationen aufzubewahren.
- Trigger können in der übrigen Tabelle oder in anderen Tabellen Daten auf Datenwerte hin überprüfen. Dies ist nützlich, wenn Sie aufgrund von Verweisen auf Daten anderer Zeilen dieser oder anderer Tabellen keine referenziellen Integritätsbedingungen oder Prüfungen auf Integritätsbedingungen verwenden können.
- Trigger können benutzerdefinierte Funktionen zur Aktivierung von anderen Operationen als Datenbankoperationen verwenden. Dies ist beispielsweise beim Absetzen von Alerts oder beim Aktualisieren von Informationen außerhalb der Datenbank nützlich.

Beispiel

Im folgenden Beispiel wird ein AFTER-Trigger dargestellt, mit dem die Anzahl der Mitarbeiter vergrößert wird, wenn ein neuer Mitarbeiter eingestellt wird.

```
CREATE TRIGGER NEW_HIRE
  AFTER INSERT ON EMPLOYEE
  FOR EACH ROW
  UPDATE COMPANY_STATS SET NBEMP = NBEMP + 1
```

INSTEAD OF-Trigger

INSTEAD OF-Trigger beschreiben die Vorgehensweise beim Durchführen von Einfüge-, Aktualisierung- und Löschoperationen für komplexe Sichten. Mit INSTEAD OF-Triggern können Anwendungen eine Sicht als alleinige Schnittstelle für alle SQL-Operationen (INSERT, DELETE, UPDATE und SELECT) verwenden.

In der Regel enthalten INSTEAD OF-Trigger die Umkehrung der in einem Hauptteil der Sicht angewendeten Logik. Beispiel: Betrachten Sie eine Sicht, bei der Spal-

ten der Quellentabelle entschlüsselt werden. Der INSTEAD OF-Trigger dieser Sicht verschlüsselt Daten und fügt sie anschließend in die Quellentabelle ein, um so die symmetrische Operation auszuführen.

Mit einem INSTEAD OF-Trigger wird die angeforderte Änderungsoperation für die Sicht durch die Triggerlogik ersetzt. Diese führt die Operation im Auftrag der Sicht durch. Aus der Sicht der Anwendung geschieht dies transparent, da sie erkennt, dass alle Operationen für die Sicht durchgeführt werden. Es ist nur ein einziger INSTEAD OF-Trigger für jede Art von Operation in einer angegebenen Themensicht zulässig.

Bei der Sicht selbst muss es sich um eine Sicht ohne Typ oder einen Aliasnamen handeln, der in eine Sicht ohne Typ aufgelöst wird. Außerdem darf es keine Sicht sein, die mit WITH CHECK OPTION (symmetrische Sicht) definiert ist, und auch keine Sicht, für die direkt oder indirekt eine symmetrische Sicht definiert wurde.

Beispiel

Im folgenden Beispiel werden drei INSTEAD OF-Trigger dargestellt, die Logik für INSERT-, UPDATE- und DELETE-Operationen für die definierte Sicht (EMPV) bereitstellen. Die Sicht EMPV enthält einen Join in ihrer Klausel FROM; deshalb ist eine native Unterstützung von Änderungsoperationen nicht möglich.

```
CREATE VIEW EMPV(EMPNO, FIRSTNME, MIDINIT, LASTNAME, PHONENO,
                 HIREDATE, DEPTNAME)
AS SELECT EMPNO, FIRSTNME, MIDINIT, LASTNAME, PHONENO,
          HIREDATE, DEPTNAME
          FROM EMPLOYEE, DEPARTMENT WHERE
               EMPLOYEE.WORKDEPT = DEPARTMENT.DEPTNO

CREATE TRIGGER EMPV_INSERT INSTEAD OF INSERT ON EMPV
REFERENCING NEW AS NEWEMP FOR EACH ROW
INSERT INTO EMPLOYEE (EMPNO, FIRSTNME, MIDINIT, LASTNAME,
                      WORKDEPT, PHONENO, HIREDATE)
        VALUES(EMPNO, FIRSTNME, MIDINIT, LASTNAME,
               COALESCE((SELECT DEPTNO FROM DEPARTMENT AS D
                         WHERE D.DEPTNAME = NEWEMP.DEPTNAME),
                        RAISE_ERROR('70001', 'Unknown dept name')),
               PHONENO, HIREDATE)

CREATE TRIGGER EMPV_UPDATE INSTEAD OF UPDATE ON EMPV
REFERENCING NEW AS NEWEMP OLD AS OLDEMP
  FOR EACH ROW
BEGIN ATOMIC
 VALUES(CASE WHEN NEWEMP.EMPNO = OLDEMP.EMPNO THEN 0
             ELSE RAISE_ERROR('70002', 'Must not change EMPNO') END);
 UPDATE EMPLOYEE AS E
   SET (FIRSTNME, MIDINIT, LASTNAME, WORKDEPT, PHONENO, HIREDATE)
     = (NEWEMP.FIRSTNME, NEWEMP.MIDINIT, NEWEMP.LASTNAME,
               COALESCE((SELECT DEPTNO FROM DEPARTMENT AS D
                         WHERE D.DEPTNAME = NEWEMP.DEPTNAME),
                        RAISE_ERROR ('70001', 'Unknown dept name')),
               NEWEMP.PHONENO, NEWEMP.HIREDATE)
 WHERE NEWEMP.EMPNO = E.EMPNO;
END

CREATE TRIGGER EMPV_DELETE INSTEAD OF DELETE ON EMPV
REFERENCING OLD AS OLDEMP FOR EACH ROW
DELETE FROM EMPLOYEE AS E WHERE E.EMPNO = OLDEMP.EMPNO
```

Entwerfen von Triggern

Wenn Sie einen Trigger erstellen, müssen Sie ihn einer Tabelle zuordnen. Handelt es sich um einen INSTEAD OF-Trigger, müssen Sie ihn einer Sicht zuordnen. Diese Tabelle bzw. Sicht wird als *Zieltabelle* des Triggers bezeichnet. Der Begriff *Änderungsoperation* bezieht sich auf jede Änderung am Status der Zieltabelle.

Informationen zu diesem Vorgang

Eine *Änderungsoperation* wird durch folgende Elemente eingeleitet:

- Anweisung INSERT
- Anweisung UPDATE bzw. referenzielle Integritätsbedingung, die eine UPDATE-Operation ausführt
- Anweisung DELETE bzw. referenzielle Integritätsbedingung, die eine DELETE-Operation ausführt
- Anweisung MERGE

Sie müssen jeden Trigger einem dieser drei Typen von Änderungsoperation zuordnen. Diese Zuordnung wird als *Trigger-Ereignis* für diesen bestimmten Trigger bezeichnet.

Sie müssen außerdem die Aktion definieren, die als *ausgelöste Aktion* bezeichnet wird und vom Trigger ausgeführt wird, wenn das Trigger-Ereignis eintritt. Die ausgelöste Aktion besteht aus einer oder mehreren Anweisungen, die entweder vor oder nach der Ausführung des Trigger-Ereignisses durch den Datenbankmanager ausgeführt werden können. Wenn ein Trigger-Ereignis eintritt, bestimmt der Datenbankmanager die Gruppe von Zeilen in der Subjekttabelle, die von der Änderungsoperation betroffen ist, und führt den Trigger aus.

Richtlinien zur Erstellung von Triggern:

Bei der Erstellung eines Triggers müssen die folgenden Attribute bzw. das folgende Verhalten deklariert werden:

- Der Name des Triggers
- Der Name der Subjekttabelle
- Die Aktivierungszeit des Triggers (vor (BEFORE) oder nach (AFTER) Ausführung der Änderungsoperation)
- Das Trigger-Ereignis (INSERT, DELETE oder UPDATE)
- Der alte Wert der Übergangsvariablen, falls vorhanden
- Der neue Wert der Übergangsvariablen, falls vorhanden
- Der alte Wert der Übergangstabelle, falls vorhanden
- Der neue Wert der Übergangstabelle, falls vorhanden
- Die Granularität (FOR EACH STATEMENT oder FOR EACH ROW)
- Die ausgelöste Aktion des Triggers (einschließlich einer Bedingung für die ausgelöste Aktion und ausgelöster Anweisung(en))
- Wenn das Trigger-Ereignis eine UPDATE-Operation ist: Eine Trigger-Spaltenliste, falls der Trigger nur aktiviert werden soll, wenn bestimmte Spalten in der Anweisung UPDATE angegeben werden

Entwerfen mehrerer Trigger:

Wenn Trigger mithilfe der Anweisung CREATE TRIGGER definiert werden, wird ihre Erstellungszeit in der Datenbank in Form einer Zeitmarke registriert. Der Wert dieser Zeitmarke wird nachfolgend dazu verwendet, die Reihenfolge der Aktivierung von Triggern zu bestimmen, wenn mehrere

Trigger vorhanden sind, die zur gleichen Zeit ausgeführt werden sollen. Die Zeitmarke wird zum Beispiel verwendet, wenn mehr als ein Trigger für dieselbe Subjekttabelle mit demselben Ereignis und derselben Aktivierungszeit vorhanden ist. Die Zeitmarke wird auch verwendet, wenn mindestens ein AFTER- oder INSTEAD OF-Trigger vorhanden ist, der durch das Trigger-Ereignis und durch Aktionen von referenziellen Integritätsbedingungen, die von der ausgelösten Aktion direkt oder indirekt verursacht werden, aktiviert wird (d. h. rekursiv durch andere referenzielle Integritätsbedingungen).

Betrachten Sie die folgenden beiden Trigger:

```
CREATE TRIGGER NEW_HIRED
  AFTER INSERT ON EMPLOYEE
  FOR EACH ROW
  BEGIN ATOMIC
    UPDATE COMPANY_STATS
    SET NBEMP = NBEMP + 1;
  END

CREATE TRIGGER NEW_HIRED_DEPT
   AFTER INSERT ON EMPLOYEE
   REFERENCING NEW AS EMP
   FOR EACH ROW
     BEGIN ATOMIC
       UPDATE DEPTS
       SET NBEMP = NBEMP + 1
       WHERE DEPT_ID = EMP.DEPT_ID;
     END
```

Die oben gezeigten Trigger werden aktiviert, wenn Sie eine INSERT-Operation an der Tabelle EMPLOYEE ausführen. In diesem Fall definiert die Zeitmarke ihrer Erstellung, welcher der beiden Trigger zuerst aktiviert wird.

Die Aktivierung der Trigger erfolgt in aufsteigender Reihenfolge der Zeitmarkenwerte. Das heißt, ein Trigger, der einer Datenbank neu hinzugefügt wurde, wird nach allen anderen, zuvor definierten Triggern ausgeführt.

Alte Trigger werden vor neuen Triggern aktiviert, um sicherzustellen, dass neue Trigger als *inkrementelle* Zusätze zu Änderungen verwendet werden können, die die Datenbank betreffen. Wenn eine ausgelöste Anweisung von Trigger T1 zum Beispiel eine neue Zeile in Tabelle T einfügt, kann eine ausgelöste Anweisung von Trigger T2, der nach T1 ausgeführt wird, dazu verwendet werden, dieselbe Zeile in T mit bestimmten Werten zu aktualisieren. Da die Aktivierungsreihenfolge von Triggern vorhersagbar ist, können Sie mehrere Trigger für eine Tabelle definieren und trotzdem sicher sein, dass die neueren eine Tabelle bearbeiten, die bereits von den älteren geändert wurde.

Interaktionen von Triggern mit referenziellen Integritätsbedingungen:
Ein Trigger-Ereignis kann infolge von Änderungen eintreten, die auf die Umsetzung von referenziellen Integritätsbedingungen zurückzuführen sind. Betrachten Sie ein Beispiel mit den beiden Tabellen DEPT und EMP: Wenn Lösch- oder Aktualisierungsoperationen an DEPT die Weitergabe von Löschungen bzw. Aktualisierungen an EMP über referenzielle Integritätsbedingungen verursachen, werden für EMP bei DELETE- oder UPDATE-Operationen definierte Trigger infolge der referenziellen Integritätsbedingungen aktiviert, die für DEPT definiert sind. Die Trigger für EMP werden entweder vor oder nach der Löschung (bei Angabe von ON DELE-

TE CASCADE) oder der Aktualisierung von Zeilen in EMP (bei Angabe von ON DELETE SET NULL) abhängig von ihrer Aktivierungszeit ausgeführt.

Angeben der Auslösebedingungen eines Triggers (auslösende Anweisung oder Auslöseereignis)

Jeder Trigger ist einem Ereignis zugeordnet. Trigger werden aktiviert, wenn das ihnen zugeordnete Ereignis in der Datenbank eintritt. Dieses Trigger-Ereignis (Auslöseereignis) tritt ein, wenn die angegebene Aktion, das heißt eine Anweisung UPDATE, INSERT oder DELETE (einschließlich solcher, die durch Aktionen von referenziellen Integritätsbedingungen verursacht werden) an der Zieltabelle ausgeführt wird.

Informationen zu diesem Vorgang

Beispiel:

```
CREATE TRIGGER NEW_HIRE
  AFTER INSERT ON EMPLOYEE
  FOR EACH ROW       UPDATE COMPANY_STATS SET NBEMP = NBEMP + 1
```

Die oben gezeigte Anweisung definiert den Trigger NEW_HIRE, der aktiviert wird, wenn Sie eine INSERT-Operation an der Tabelle EMPLOYEE ausführen.

Sie ordnen jedes Trigger-Ereignis und infolgedessen jeden Trigger genau einer Zieltabelle und genau einer Änderungsoperation zu. Folgende Änderungsoperationen sind verfügbar:

INSERT-Operation
Eine INSERT-Operation kann nur durch eine Anweisung INSERT oder durch die INSERT-Operation einer Anweisung MERGE verursacht werden. Daher werden entsprechende Trigger nicht aktiviert, wenn Daten mithilfe von Dienstprogrammen geladen werden, die kein INSERT verwenden, wie zum Beispiel der Befehl **LOAD**.

DELETE-Operation
Eine DELETE-Operation kann durch eine Anweisung DELETE, durch die DELETE-Operation einer Anweisung MERGE oder durch die referenzielle Integritätsregel ON DELETE CASCADE verursacht werden.

UPDATE-Operation
Eine UPDATE-Operation kann durch eine Anweisung UPDATE, durch die UPDATE-Operation einer Anweisung MERGE oder durch die referenzielle Integritätsregel ON DELETE SET NULL verursacht werden.

Wenn das Trigger-Ereignis eine UPDATE-Operation ist, kann das Ereignis mit bestimmten Spalten der Zieltabelle verknüpft werden. In diesem Fall wird der Trigger nur aktiviert, wenn die UPDATE-Operation versucht, eine der angegebenen Spalten zu aktualisieren. Dies bietet weitere Verfeinerungsmöglichkeiten für die Angabe des Ereignisses, das den Trigger aktiviert.

Zum Beispiel wird der folgende Trigger REORDER nur aktiviert, wenn eine UPDATE-Operation an den Spalten ON_HAND oder MAX_STOCKED der Tabelle PARTS ausgeführt wird:

```
CREATE TRIGGER REORDER
  AFTER UPDATE OF ON_HAND, MAX_STOCKED ON PARTS
  REFERENCING NEW AS N_ROW
  FOR EACH ROW       WHEN (N_ROW.ON_HAND < 0.10 * N_ROW.MAX_STOCKED)
```

```
BEGIN ATOMIC
VALUES(ISSUE_SHIP_REQUEST(N_ROW.MAX_STOCKED -
                          N_ROW.ON_HAND,
                          N_ROW.PARTNO));
END
```

Wenn ein Trigger aktiviert wird, wird er entsprechend seiner Granularität wie folgt ausgeführt:

FOR EACH ROW
: Er wird so oft ausgeführt, wie Zeilen in der Menge der betroffenen Zeilen enthalten sind. Wenn Sie sich auf die bestimmten, von der ausgelösten Aktion betroffenen Zeilen beziehen müssen, verwenden Sie die Granularität FOR EACH ROW. Ein Beispiel für diesen Fall ist der Vergleich der neuen und alten Werte einer aktualisierten Zeile in einem mit AFTER UPDATE definierten Trigger.

FOR EACH STATEMENT
: Der Trigger wird einmal für das gesamte Trigger-Ereignis ausgeführt.

Wenn die Menge der betroffenen Zeilen leer ist (d. h. im Fall einer gezielten UPDATE- oder DELETE-Operation, bei der die WHERE-Klausel keine Zeilen ergeben hat), wird der Trigger mit der Definition FOR EACH ROW nicht ausgeführt. Ein Trigger mit der Definition FOR EACH STATEMENT wird jedoch auch in diesem Fall einmal ausgeführt.

Zum Beispiel kann eine Verfolgung der Anzahl von Mitarbeitern mit der Granularität FOR EACH ROW realisiert werden:

```
CREATE TRIGGER NEW_HIRED
  AFTER INSERT ON EMPLOYEE
  FOR EACH ROW
  UPDATE COMPANY_STATS SET NBEMP = NBEMP + 1
```

Denselben Effekt können Sie auch mit einer Aktualisierung erzielen, indem Sie die Granularität FOR EACH STATEMENT verwenden:

```
CREATE TRIGGER NEW_HIRED
  AFTER INSERT ON EMPLOYEE
  REFERENCING NEW_TABLE AS NEWEMPS
  FOR EACH STATEMENT
  UPDATE COMPANY_STATS
  SET NBEMP = NBEMP + (SELECT COUNT(*) FROM NEWEMPS)
```

Anmerkung:

- Die Granularität FOR EACH STATEMENT wird von Vortriggern (BEFORE-Triggern) nicht unterstützt.
- Die maximale Verschachtelungsebene für Trigger beträgt 16. Das heißt, die maximale Anzahl hintereinandergeschalteter Triggeraktivierungen beträgt 16. Eine Triggeraktivierung bezieht sich auf die Aktivierung eines Triggers aufgrund eines auslösenden Ereignisses, wie z. B. einer Einfügung, einer Aktualisierung oder einer Löschung von Daten in einer Tabellenspalte, oder im Allgemeinen auf eine Tabelle.

Angeben des Aktivierungszeitpunkts eines Triggers (Klauseln BEFORE, AFTER und INSTEAD OF)

Die *Aktivierungszeit des Triggers* gibt an, wann der Trigger in Relation zum Trigger-Ereignis aktiviert werden soll.

Informationen zu diesem Vorgang

Es gibt drei Aktivierungszeiten, die Sie angegeben können: BEFORE, AFTER oder INSTEAD OF:

- Wenn die Aktivierungszeit mit BEFORE (Vortrigger) definiert wird, werden die ausgelösten Aktionen für jede Zeile in der Menge der betroffenen Zeilen aktiviert, bevor das Trigger-Ereignis ausgeführt wird. Das bedeutet, dass die Subjekttabelle erst geändert wird, nachdem der Vortrigger (BEFORE-Trigger) die Ausführung seiner Aktion für jede Zeile abgeschlossen hat. Beachten Sie, dass Vortrigger die Granularität FOR EACH ROW haben müssen.
- Wenn die Aktivierungszeit mit AFTER (Nachtrigger) definiert wird, werden die ausgelösten Aktionen entsprechend der angegebenen Granularität des Triggers für jede Zeile in der Menge der betroffenen Zeilen oder für die Anweisung aktiviert. Dies geschieht, nachdem das Trigger-Ereignis ausgeführt wurde und der Datenbankmanager alle Integritätsbedingungen, die von dem Trigger-Ereignis betroffen sein könnten, einschließlich der Aktionen durch referenzielle Integritätsbedingungen, überprüft hat. Beachten Sie, dass für Nachtrigger Granularität FOR EACH ROW oder FOR EACH STATEMENT angegeben werden kann.

  Die Aktivierungszeit des folgenden Triggers ist zum Beispiel nach der INSERT-Operation an der Tabelle EMPLOYEE:

  ```
  CREATE TRIGGER NEW_HIRE
    AFTER INSERT ON EMPLOYEE
    FOR EACH ROW
    UPDATE COMPANY_STATS SET NBEMP = NBEMP + 1
  ```

- Wenn die Aktivierungszeit mit INSTEAD OF definiert wird, werden die ausgelösten Aktionen für jede Zeile in der Menge der betroffenen Zeilen anstatt der Ausführung des Trigger-Ereignisses aktiviert. INSTEAD OF-Trigger müssen die Granularität FOR EACH ROW haben, und als Subjekttabelle muss eine Sicht angegeben werden. Keine anderen Trigger können eine Sicht als Subjekttabelle verwenden.

Beispiel

Das folgende Diagramm veranschaulicht das Ausführungsmodell von Vortriggern und Nachtriggern:

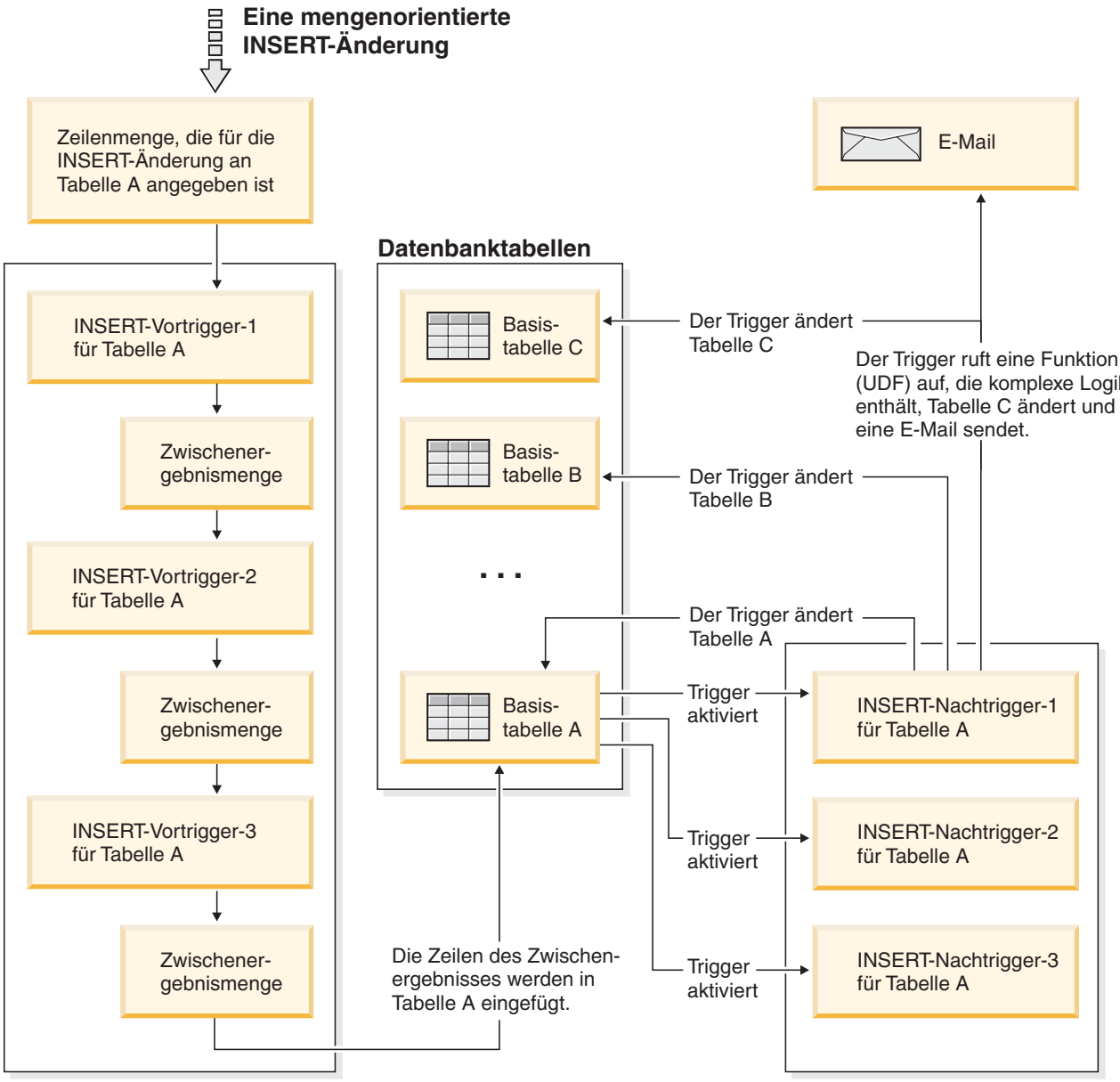


Abbildung 48. Ausführungsmodell für Trigger

Für eine Tabelle mit Vortriggern und Nachtriggern sowie mit einem Änderungsereignis, das diesen Triggern zugeordnet ist, werden alle Vortrigger zuerst aktiviert. Der erste Vortrigger für ein bestimmtes Ereignis operiert an der Menge der Zeilen, die Ziel der Operation sind, und nimmt alle Aktualisierungsänderungen an der Menge vor, die von seiner Logik vorgeschrieben werden. Die Ausgabe dieses Vortriggers wird vom nächsten Vortrigger als Eingabe empfangen. Wenn alle Vortrigger, die von dem Ereignis aktiviert wurden, ausgeführt wurden, wird die Zwischenergebnismenge, das heißt das Ergebnis der Änderungen durch die Vortrigger an den Zielzeilen der Trigger-Ereignisoperation, auf die Tabelle angewendet. Nachfolgend werden die einzelnen, dem Ereignis zugeordneten Nachtrigger aktiviert. Die Nachtrigger können dieselbe Tabelle oder eine andere Tabelle ändern oder auch eine datenbankexterne Aktion ausführen.

Die verschiedenen Aktivierungszeiten von Triggern entsprechen den verschiedenen Zwecken von Triggern. Im Grunde sind Vortrigger eine Erweiterung des Subsys-

tems der Integritätsbedingungen des Datenbankmanagementsystems. Daher werden sie in der Regel zu folgenden Zwecken eingesetzt:

- Zur Überprüfung von Eingabedaten
- Zur automatischen Generierung von Werten für neu eingefügte Zeilen
- Zum Lesen aus anderen Tabellen zur Auflösung von Querverweisen

Vortrigger werden nicht zur weiteren Modifikation der Datenbank verwendet, weil sie aktiviert werden, bevor das Trigger-Ereignis auf die Datenbank angewendet wird. Infolgedessen werden sie aktiviert, bevor Integritätsbedingungen überprüft werden.

Im Gegensatz dazu können Sie Nachtrigger als Modul der Anwendungslogik betrachten, das in der Datenbank bei jedem Auftreten eines bestimmten Ereignisses ausgeführt wird. Als Teil einer Anwendung sehen Nachtrigger die Datenbank immer in einem konsistenten Status. Beachten Sie, dass diese Trigger nach den Überprüfungen der Integritätsbedingungen ausgeführt werden. Infolgedessen können Sie sie in erster Linie zur Ausführung von Operationen verwenden, die auch von einer Anwendung ausgeführt werden können. Beispiel:

- Sie können Folgeänderungsoperationen in der Datenbank ausführen.
- Sie können Aktionen außerhalb der Datenbank ausführen, zum Beispiel um Alerts zu unterstützen. Beachten Sie, dass Aktionen, die außerhalb der Datenbank ausgeführt werden, nicht rückgängig gemacht werden, falls der Trigger rückgängig gemacht wird.

Im Unterschied dazu können Sie einen INSTEAD OF-Trigger als Beschreibung der inversen Operation der Sicht, für die er definiert ist, betrachten. Wenn zum Beispiel die SELECT-Liste in der Sicht einen Ausdruck über eine Tabelle enthält, enthält die INSERT-Anweisung im Hauptteil des entsprechenden INSTEAD OF-Triggers für die INSERT-Operation den umgekehrten Ausdruck.

Aufgrund der unterschiedlichen Merkmale von BEFORE-, AFTER- und INSTEAD OF-Triggern kann jeweils eine andere Gruppe von SQL-Operationen zur Definition der ausgelösten Aktionen von BEFORE-, AFTER- und INSTEAD OF-Triggern verwendet werden. So sind zum Beispiel UPDATE-Operationen in Vortriggern nicht zulässig, da es keine Garantie gibt, dass keine Integritätsbedingungen durch die ausgelöste Aktion verletzt werden. Analog werden in BEFORE-, AFTER- und INSTEAD OF-Triggern verschiedene Triggergranularitäten unterstützt.

Die ausgelöste SQL-Anweisung aller Trigger kann eine dynamische zusammengesetzte Anweisung sein. Für Vortrigger gelten jedoch einige Einschränkungen, da sie die folgenden SQL-Anweisungen nicht enthalten dürfen:

- UPDATE
- DELETE
- INSERT
- MERGE

Definieren von Bedingungen für den Aktivierungszeitpunkt einer Triggeraktion (Klausel WHEN)

Die Aktivierung eines Triggers führt zur Ausführung der zugeordneten ausgelösten Aktion. Jeder Trigger hat genau eine ausgelöste Aktion, die wiederum zwei Komponenten hat: eine optionale *Bedingung zum Auslösen der Aktion* bzw. WHEN-Klausel und eine Folge aus einer oder mehreren *ausgelösten Anweisungen*.

Informationen zu diesem Vorgang

Die *Bedingung zum Auslösen der Aktion* ist eine optionale Klausel der ausgelösten Aktion, die eine Suchbedingung angibt, die als wahr ausgewertet werden muss, damit die Anweisungen in der ausgelösten Aktion ausgeführt werden. Wenn die WHEN-Klausel nicht angegeben wird, werden die Anweisungen in der ausgelösten Aktion in jedem Fall ausgeführt.

Die Bedingung zum Auslösen der Aktion wird einmal pro Zeile ausgewertet, wenn der Trigger mit der Granularität FOR EACH ROW definiert ist, und einmal pro Anweisung, wenn der Trigger mit der Granularität FOR EACH STATEMENT definiert ist.

Diese Klausel bietet weitere Steuerungsmöglichkeiten für die Feinabstimmung der Aktionen, die durch einen Trigger aktiviert werden. Ein Beispiel für die Nützlichkeit der WHEN-Klausel ist die Umsetzung einer von den Daten abhängigen Regel, bei der die ausgelöste Aktion nur aktiviert wird, wenn der eingehende Wert innerhalb oder außerhalb eines bestimmten Bereichs liegt.

Die Aktivierung eines Triggers führt zur Ausführung der zugeordneten ausgelösten Aktion. Jeder Trigger hat genau eine ausgelöste Aktion, die wiederum zwei Komponenten hat: eine optionale Bedingung zum Auslösen der Aktion bzw. WHEN-Klausel und eine Folge aus einer oder mehreren ausgelösten Anweisungen.

Die Bedingung zum Auslösen der Aktion definiert, ob die ausgelösten Anweisungen für die Zeile bzw. für die Anweisung, für die die ausgelöste Aktion ausgeführt wird, ausgeführt werden oder nicht. Die ausgelösten Anweisungen definieren die Aktionen, die von dem Trigger in der Datenbank ausgeführt werden, wenn das dem Trigger zugeordnete Ereignis eintritt.

Beispiel

Die folgende Triggeraktion gibt zum Beispiel an, dass die ausgelösten Anweisungen nur für Zeilen aktiviert werden sollen, in denen der Wert der Spalte ON_HAND kleiner als zehn Prozent des Werts der Spalte MAX_STOCKED ist. In diesem Fall wird durch die ausgelösten Anweisungen die Funktion ISSUE_SHIP_REQUEST aufgerufen.

```
CREATE TRIGGER REORDER
  AFTER UPDATE OF ON_HAND, MAX_STOCKED ON PARTS
  REFERENCING NEW AS N_ROW
  FOR EACH ROW

  WHEN (N_ROW.ON_HAND < 0.10 * N_ROW.MAX_STOCKED)
  BEGIN ATOMIC
    VALUES(ISSUE_SHIP_REQUEST(N_ROW.MAX_STOCKED -
                              N_ROW.ON_HAND,
                              N_ROW.PARTNO));
  END
```

Die ausgelösten Anweisungen führen die tatsächlichen Aktionen aus, die durch die Aktivierung eines Triggers verursacht werden. Nicht jede SQL-Operation ist in jeder Art von Trigger sinnvoll. Je nachdem, ob die Aktivierungszeit des Triggers mit BEFORE oder AFTER definiert ist, eignen sich verschiedene Arten von Operationen als ausgelöste Anweisungen.

Wenn eine der ausgelösten Anweisungen einen negativen Rückkehrcode zurückgibt, werden in den meisten Fällen die auslösende Anweisung sowie alle Triggeraktionen und Aktionen durch referenzielle Integritätsbedingungen rückgängig ge-

macht (ROLLBACK). Der Name des Triggers, der SQLCODE-Wert und der SQLSTATE-Wert sowie viele der Token aus der fehlgeschlagenen ausgelösten Anweisung werden in der Fehlernachricht zurückgegeben.

Unterstützte SQL PL-Anweisungen in Triggern

Die ausgelöste SQL-Anweisung aller Trigger kann eine zusammengesetzte, dynamische Anweisung sein.

Das heißt, ausgelöste SQL-Anweisungen können eines oder mehrere der folgenden Elemente enthalten:

- Anweisung CALL
- Anweisung DECLARE variable
- Anweisung SET variable
- WHILE-Schleife
- FOR-Schleife
- Anweisung IF
- Anweisung SIGNAL
- Anweisung ITERATE
- Anweisung LEAVE
- Anweisung GET DIGNOSTIC
- Fullselect

Jedoch können nur Nachtrigger (AFTER-Trigger) und INSTEAD OF-Trigger eine oder mehrere der folgenden SQL-Anweisungen enthalten:

- Anweisung UPDATE
- Anweisung DELETE
- Anweisung INSERT
- Anweisung MERGE

Zugreifen auf alte und neue Spaltenwerte in Triggern durch Übergangsvariablen

Wenn Sie einen Trigger mit der Definition FOR EACH ROW implementieren, kann es erforderlich sein, auf den Wert der Spalten der Zeile in der Menge der betroffenen Zeilen, für die der Trigger gerade ausgeführt wird, zu verweisen. Beachten Sie, dass Sie zum Verweisen auf Spalten in Tabellen in der Datenbank (einschließlich der Subjekttabelle) reguläre SELECT-Anweisungen verwenden können.

Informationen zu diesem Vorgang

Ein mit der Granularität FOR EACH ROW definierter Trigger kann auf Spalten der Zeile verweisen, für die er gerade ausgeführt wird, indem er zwei Übergangsvariablen verwendet, die in der Klausel REFERENCING einer Anweisung CREATE TRIGGER angegeben werden. Es gibt zwei Arten von Übergangsvariablen, die durch OLD und NEW sowie einen Korrelationsnamen angegeben werden. Sie besitzen folgende Semantik:

OLD AS korrelationsname
: Gibt einen Korrelationsnamen an, der den ursprünglichen Status der Zeile erfasst, das heißt, bevor die ausgelöste Aktion auf die Datenbank angewendet wird.

NEW AS korrelationsname
Gibt einen Korrelationsnamen an, der den Wert erfasst, der zum Aktualisieren der Zeile in der Datenbank verwendet wird bzw. wurde, wenn die ausgelöste Aktion auf die Datenbank angewendet wird.

Beispiel

Betrachten Sie das folgende Beispiel:

```
CREATE TRIGGER REORDER
  AFTER UPDATE OF ON_HAND, MAX_STOCKED ON PARTS
  REFERENCING NEW AS N_ROW
  FOR EACH ROW
  WHEN (N_ROW.ON_HAND < 0.10 * N_ROW.MAX_STOCKED
  AND N_ROW.ORDER_PENDING = 'N')
  BEGIN ATOMIC
    VALUES(ISSUE_SHIP_REQUEST(N_ROW.MAX_STOCKED -
                              N_ROW.ON_HAND,
                              N_ROW.PARTNO));
    UPDATE PARTS SET PARTS.ORDER_PENDING = 'Y'
    WHERE PARTS.PARTNO = N_ROW.PARTNO;
  END
```

Nächste Schritte

An der oben gezeigten Definition der Übergangsvariablen OLD und NEW wird klar, dass nicht jede Übergangsvariable für jeden Trigger definiert werden kann. Übergangsvariablen können abhängig von der Art des Trigger-Ereignisses definiert werden:

UPDATE
Ein UPDATE-Trigger kann beide Übergangsvariablen (OLD und NEW) verwenden.

INSERT
Ein INSERT-Trigger kann sich nur auf eine Übergangsvariable NEW beziehen, da die betroffene Zeile vor der Aktivierung der INSERT-Operation in der Datenbank nicht vorhanden ist. Das heißt, dass es keinen ursprünglichen Status der Zeile gibt, der alte Werte definieren würde, bevor die ausgelöste Aktion auf die Datenbank angewendet wird.

DELETE
Ein DELETE-Trigger kann sich nur auf eine Übergangsvariable OLD beziehen, weil in der DELETE-Operation keine neuen Werte angegeben werden.

Anmerkung: Übergangsvariablen können nur für Trigger angegeben werden, die mit der Granularität FOR EACH ROW definiert werden. In einem Trigger mit der Granularität FOR EACH STATEMENT reicht ein Verweis auf eine Übergangsvariable nicht aus, um anzugeben, auf welche der Zeilen in der Menge der betroffenen Zeilen sich die Übergangsvariable bezieht. Verweisen Sie stattdessen auf die Menge der alten und neuen Zeilen, indem Sie die Klauseln OLD TABLE und NEW TABLE der Anweisung CREATE TRIGGER verwenden. Weitere Informationen zu diesen Klauseln finden Sie in der Beschreibung der Anweisung CREATE TRIGGER.

Verweisen auf alte und neue Tabellenergebnismengen mit Übergangstabellen

In Triggern, die mit FOR EACH ROW bzw. FOR EACH STATEMENT definiert sind, kann es erforderlich sein, auf die gesamte Menge der betroffenen Zeilen zu

verweisen. Dies ist zum Beispiel der Fall, wenn der Triggerhauptteil Spaltenberechnungen auf die Menge der betroffenen Zeilen anwenden muss (z. B. MAX, MIN oder AVG für einige Spaltenwerte).

Informationen zu diesem Vorgang

Ein Trigger kann auf die Menge der betroffenen Zeilen Bezug nehmen, indem er zwei Übergangstabellen verwendet, die in der Klausel REFERENCING einer Anweisung CREATE TRIGGER angegeben werden können. Ebenso wie bei Übergangsvariablen gibt es auch bei Übergangstabellen zwei Arten, die durch OLD_TABLE und NEW_TABLE sowie durch einen Tabellennamen mit der folgenden Semantik angegeben werden:

OLD_TABLE AS tabellenname
Gibt den Namen der Tabelle an, die den ursprünglichen Status der Menge der betroffenen Zeilen erfasst (d. h. bevor die auslösende SQL-Operation auf die Datenbank angewendet wird).

NEW_TABLE AS tabellenname
Gibt den Namen der Tabelle an, die den Wert erfasst, der zur Aktualisierung der Zeilen in der Datenbank verwendet wird, wenn die ausgelöste Aktion auf die Datenbank angewendet wird.

Beispiel

Beispiel:

```
CREATE TRIGGER REORDER
  AFTER UPDATE OF ON_HAND, MAX_STOCKED ON PARTS
  REFERENCING NEW_TABLE AS N_TABLE
  NEW AS N_ROW
  FOR EACH ROW
  WHEN ((SELECT AVG (ON_HAND) FROM N_TABLE) > 35)
  BEGIN ATOMIC
    VALUES(INFORM_SUPERVISOR(N_ROW.PARTNO,
                             N_ROW.MAX_STOCKED,
                             N_ROW.ON_HAND));
  END
```

Beachten Sie, dass NEW_TABLE immer die vollständige Menge der aktualisierten Zeilen enthält, auch für einen Trigger mit der Granularität FOR EACH ROW. Wenn ein Trigger Operationen an der Tabelle ausführt, für die der Trigger definiert ist, enthält NEW_TABLE die geänderten Zeilen aus der Anweisung, die den Trigger aktiviert hat. Die Übergangstabelle NEW_TABLE enthält jedoch nicht die geänderten Zeilen, die durch Anweisungen innerhalb des Triggers bewirkt wurden, da dies wiederum eine separate Aktivierung des Triggers zur Folge hätte.

Nächste Schritte

Die Übergangstabellen sind schreibgeschützt. Für Übergangstabellen gelten die gleichen Regeln, die auch bei der Definition der Art von Übergangsvariablen gelten, die für die verschiedenen Trigger-Ereignisse definiert werden können:

UPDATE
Ein UPDATE-Trigger kann beide Übergangstabellen (OLD_TABLE und NEW_TABLE) verwenden.

INSERT
Ein INSERT-Trigger kann sich nur auf eine Übergangstabelle NEW_TABLE beziehen, da die betroffenen Zeilen vor der Aktivierung der INSERT-Ope-

ration in der Datenbank nicht vorhanden sind. Das heißt, dass es keinen ursprünglichen Status der Zeilen gibt, der alte Werte definiert, bevor die ausgelöste Aktion auf die Datenbank angewendet wird.

DELETE
Ein DELETE-Trigger kann sich nur auf eine Übergangstabelle OLD_TABLE beziehen, weil in der DELETE-Operation keine neuen Werte angegeben werden.

Anmerkung: Es wichtig zu beachten, dass Übergangstabellen für beide Granularitäten von AFTER-Triggern angegeben werden können: FOR EACH ROW und FOR EACH STATEMENT.

Der Geltungsbereich der Angabe `OLD_TABLE` und `NEW_TABLE tabellenname` ist der Triggerhauptteil. In diesem Geltungsbereich hat dieser Tabellenname Vorrang vor den Namen jeder anderen Tabelle mit demselben *tabellennamen* ohne Qualifikationsmerkmal, die möglicherweise im Schema vorhanden ist. Das bedeutet, wenn für `OLD_TABLE` bzw. `NEW_TABLE tabellenname` zum Beispiel 'X' angegeben wird, bezieht sich ein Verweis auf X (d. h. ein X ohne Qualifikationsmerkmal) in der FROM-Klausel einer SELECT-Anweisung immer auf die Übergangstabelle, auch wenn im Schema des Trigger-Erstellers eine Tabelle mit dem Namen 'X' vorhanden ist. In diesem Fall muss der Benutzer den vollständig qualifizierten Namen verwenden, um auf die Tabelle 'X' im Schema zu verweisen.

Erstellen von Triggern

Ein Trigger definiert eine Reihe von Aktionen, die in Verbindung mit bzw. ausgelöst durch eine Anweisung INSERT, UPDATE oder DELETE für eine angegebene Tabelle bzw. eine typisierte Tabelle ausgeführt werden.

Informationen zu diesem Vorgang

Verwenden Sie Trigger zu folgenden Zwecken:

- Überprüfen der Gültigkeit von Eingabedaten
- Generieren eines Werts für eine neu eingefügte Zeile
- Lesen aus anderen Tabellen zur Auflösung von Querverweisen
- Schreiben in andere Tabellen zur Führung von Prüfprotokollen

Trigger können zur Unterstützung allgemeiner Formen von Datenintegrität und Geschäftsregeln verwendet werden. Zum Beispiel kann ein Trigger den Kreditrahmen eines Kunden überprüfen, bevor eine Bestellung entgegengenommen oder eine Tabelle mit Übersichtsdaten aktualisiert wird.

Vorteile:

- Schnellere Anwendungsentwicklung: Da ein Trigger in der Datenbank gespeichert wird, müssen die durch den Trigger ausgeführten Aktionen nicht mehr in jeder Anwendung codiert werden.
- Einfachere Wartung: Nach dem Definieren eines Triggers wird der betreffende Trigger automatisch aufgerufen, wenn auf die Tabelle, für die er erstellt wurde, zugegriffen wird.
- Globale Implementierung von Geschäftsregeln: Wenn sich die geschäftsinternen Abläufe oder Regeln ändern, müssen lediglich die Trigger und nicht sämtliche Anwendungsprogramme geändert werden.

Beim Erstellen eines ganzheitlichen Triggers (ATOMIC) muss das Zeichen für das Anweisungsende sorgfältig behandelt werden. Der Befehlszeilenprozessor erkennt standardmäßig ein Semikolon („;") als Markierung für das Anweisungsende. Sie sollten das Zeichen für das Anweisungsende manuell im Script für die Erstellung des ganzheitlichen Triggers editieren, um ein anderes Zeichen als das Semikolon („;") zu verwenden. Das Semikolon („;") kann zum Beispiel durch ein anderes Sonderzeichen wie das Nummernzeichen („#") ersetzt werden. Sie können der DDL-Anweisung CREATE TRIGGER auch Folgendes voranstellen:

```
--#SET TERMINATOR @
```

Wenn das Abschlusszeichen im Befehlszeilenprozessor während der Verarbeitung geändert werden soll, wird es durch die folgende Syntax zurückgesetzt:

```
--#SET TERMINATOR
```

Geben Sie in die Befehlszeile die folgende Anweisung ein, um einen Trigger zu erstellen:

```
db2 -td begrenzer -vf script
```

Dabei ist *begrenzer* das alternative Zeichen für das Anweisungsende und *script* das modifizierte Script mit dem neuen *begrenzer*.

Der Hauptteil einer Triggerdefinition kann eine oder mehrere der folgenden Anweisungen enthalten: Anweisung INSERT, gezielte Anweisung UPDATE, gezielte Anweisung DELETE, Fullselect, Anweisung SET variable und Anweisung SIGNAL SQLSTATE. Der Trigger kann vor oder nach der Anweisung INSERT, UPDATE bzw. DELETE, auf die er bezogen ist, aktiviert werden.

Einschränkungen

- Trigger können nicht mit Kurznamen verwendet werden.
- Wenn es sich bei dem Trigger um einen Vortrigger (BEFORE) handelt, darf der von der ausgelösten Aktion angegebene Spaltenname keine generierte Spalte außer einer Identitätsspalte bezeichnen. Dies bedeutet, dass der generierte Identitätswert für Vortrigger sichtbar ist.

Vorgehensweise

Geben Sie in die Befehlszeile die folgende Anweisung ein, um einen Trigger zu erstellen:

```
CREATE TRIGGER name
   aktion ON tabellenname
   operation
   ausgelöste_aktion
```

Beispiel

Mit der folgenden Anweisung wird ein Trigger erstellt, der die Anzahl der Mitarbeiter automatisch erhöht, wenn eine neue Person angestellt wird, indem zum Wert in der Spalte für die Anzahl der Mitarbeiter (NBEMP) in der Tabelle COMPANY_STATS jedes Mal der Wert 1 addiert wird, wenn der Tabelle EMPLOYEE eine Zeile hinzugefügt wird:

```
CREATE TRIGGER NEW_HIRED
   AFTER INSERT ON EMPLOYEE
   FOR EACH ROW
   UPDATE COMPANY_STATS SET NBEMP = NBEMP+1;
```

Ändern und Löschen von Triggern

Trigger können nicht geändert werden. Sie müssen gelöscht und anschließend den neuen erforderlichen Definitionen entsprechend erstellt werden.

Vorbereitende Schritte

Triggerabhängigkeiten

- Alle Abhängigkeiten eines Triggers von einem anderen Objekt werden in der Katalogsicht SYSCAT.TRIGDEP aufgezeichnet. Ein Trigger kann von vielen Objekten abhängig sein.
- Wenn ein Objekt, von dem ein Trigger abhängig ist, gelöscht wird, wird der Trigger funktionsunfähig. Die Definition verbleibt jedoch in der Katalogsicht. Um einen solchen Trigger wieder funktionsfähig zu machen, müssen Sie die entsprechende Definition aus der Systemkatalogsicht abrufen und eine neue Anweisung CREATE TRIGGER ausführen.
- Wenn ein Trigger gelöscht wird (DROP), wird die entsprechende Beschreibung aus der Systemkatalogsicht SYSCAT.TRIGGERS gelöscht, und alle zugehörigen Einträge für Abhängigkeiten werden aus der Systemkatalogsicht SYSCAT.TRIGDEP gelöscht. Alle Pakete mit Abhängigkeiten über Anweisungen UPDATE, INSERT oder DELETE für den Trigger, werden inaktiviert (ungültig gemacht).
- Wenn die Sicht von dem Trigger abhängig ist und funktionsunfähig gemacht wird, wird der Trigger ebenfalls als funktionsunfähig markiert. Alle Pakete, die von Triggern abhängig sind, die als funktionsunfähig markiert wurden, werden inaktiviert.

Informationen zu diesem Vorgang

Ein Triggerobjekt kann mit der Anweisung DROP TRIGGER gelöscht werden. Dieses Verfahren hat jedoch zur Folge, dass abhängige Pakete als ungültig markiert werden, wie im Folgenden beschrieben:

- Wenn ein UPDATE-Trigger, für den keine explizite Spaltenliste definiert ist, gelöscht wird, werden Pakete, die eine Aktualisierung an der Zieltabelle ausführen, ungültig gemacht (inaktiviert).
- Wenn ein UPDATE-Trigger, für den eine Spaltenliste definiert ist, gelöscht wird, werden Pakete, die eine Aktualisierung an der Zieltabelle ausführen, nur dann ungültig gemacht, wenn das Paket auch eine Aktualisierung für mindestens eine Spalte in der Liste der Spaltennamen der Anweisung CREATE TRIGGER enthält.
- Wenn ein INSERT-Trigger gelöscht wird, werden Pakete, die eine Einfügung für die Zieltabelle enthalten, ungültig gemacht.
- Wenn ein DELETE-Trigger gelöscht wird, werden Pakete, die eine Löschung für die Zieltabelle enthalten, ungültig gemacht.

Ein Paket bleibt ungültig (inaktiviert), bis das Anwendungsprogramm explizit gebunden (Bind) bzw. erneut gebunden (Rebind) wird oder bis das Anwendungsprogramm ausgeführt wird und der Datenbankmanager einen automatischen Rebind durchführt.

Beispiele zu Triggern und zur Verwendung von Triggern

Beispiele für die Interaktion zwischen Triggern und referenziellen Integritätsbedingungen

Aktualisierungsoperationen (UPDATE) können zu Interaktionen von Triggern mit referenziellen Integritätsbedingungen und Prüfungen auf Integritätsbedingungen führen.

Abb. 40 auf Seite 462 und die zugehörige Beschreibung sind für die Verarbeitung repräsentativ, die für eine Anweisung ausgeführt wird, die Daten in der Datenbank aktualisiert.

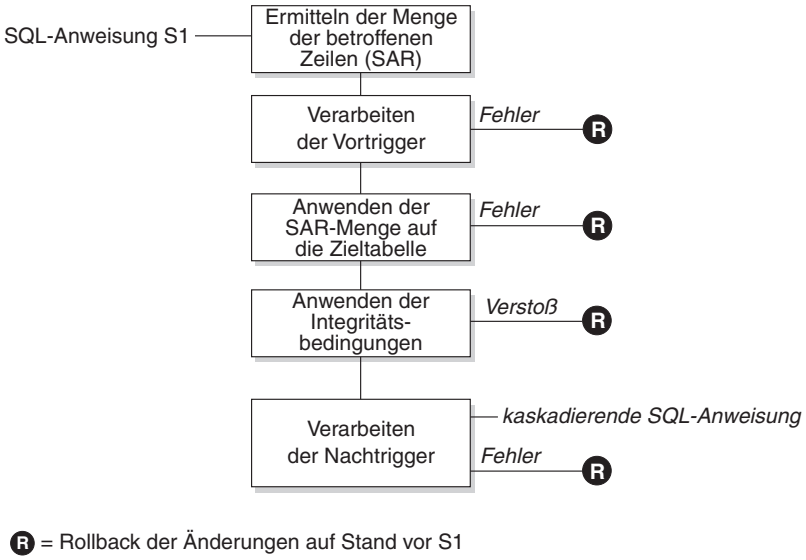


Abbildung 49. Verarbeitung einer Anweisung mit definierten Triggern und Integritätsbedingungen

Abb. 40 auf Seite 462 zeigt den allgemeinen Verarbeitungsablauf für eine Anweisung, die eine Tabelle aktualisiert. Sie geht von einer Situation aus, bei der die Tabelle Vortrigger, referenzielle Integritätsbedingungen, Prüfungen auf Integritätsbedingungen und Nachtrigger enthält, die hintereinander (kaskadierend) verarbeitet werden. Im Folgenden werden die Kästchen und anderen Elemente in Abb. 40 auf Seite 462 beschrieben.

- Anweisung S_1

  Dies ist die DELETE-, INSERT- oder UPDATE-Anweisung, die den Prozess erfordert. Die Anweisung S_1 gibt eine Tabelle (bzw. eine aktualisierbare Sicht für eine Tabelle) an, die in dieser Beschreibung als *Subjekttabelle* bezeichnet wird.
- Ermitteln der Menge der betroffenen Zeilen

  Dieser Schritt ist der Ausgangspunkt für einen Prozess, der für die Löschregeln CASCADE und SET NULL referenzieller Integritätsbedingungen sowie für kaskadierende Anweisungen aus Nachtriggern wiederholt ausgeführt wird.

  Zweck dieses Schritts ist die Ermittlung der *Menge der betroffenen Zeilen* (Set of Affected Rows - SAR) für die Anweisung. Die in dieser Menge enthaltenen Zeilen hängen von der Anweisung ab:

  – Für DELETE: Alle Zeilen, die die Suchbedingung der Anweisung erfüllen (oder die aktuelle Zeile bei einer positionierten DELETE-Operation)

– Für INSERT: Die Zeilen, die durch die Klausel VALUES oder den Fullselect angegeben werden
– Für UPDATE: Alle Zeilen, die die Suchbedingung erfüllten (oder die aktuelle Zeile bei einer positionierten UPDATE-Operation)

Wenn die Menge der betroffenen Zeilen leer ist, sind keine Vortrigger, keine Änderungen zur Anwendung auf die Subjekttabelle und keine Integritätsbedingungen für die Anweisung zu verarbeiten.

- Verarbeiten der Vortrigger

  Alle Vortrigger (BEFORE) werden in der aufsteigenden Reihenfolge ihrer Erstellung verarbeitet. Jeder Vortrigger verarbeitet die ausgelöste Aktion einmal für jede Zeile in der Menge der betroffenen Zeilen.

  Während der Verarbeitung einer ausgelösten Aktion kann ein Fehler auftreten. In diesem Fall werden alle Änderungen, die infolge der ursprünglichen Anweisung S_1 (bis dahin) ausgeführt wurden, rückgängig gemacht (Rollback).

  Wenn keine Vortrigger definiert sind oder die Menge der betroffenen Zeilen leer ist, wird dieser Schritt übersprungen.

- Anwenden der Menge der betroffenen Zeilen auf die Subjekttabelle

  Die tatsächliche DELETE-, INSERT- oder UPDATE-Operation wird unter Verwendung der Menge der betroffenen Zeilen auf die Subjekttabelle (Zieltabelle) in der Datenbank angewendet.

  Beim Anwenden der Menge der betroffenen Zeilen kann ein Fehler auftreten (z. B. wenn versucht wird, eine Zeile mit einem bereits vorhandenen Schlüssel einzufügen, obwohl ein eindeutiger Index vorhanden ist). In diesem Fall werden alle Änderungen, die infolge der ursprünglichen Anweisung S_1 (bis dahin) ausgeführt wurden, rückgängig gemacht (Rollback).

- Anwenden der Integritätsbedingungen

  Die der Subjekttabelle zugeordneten Integritätsbedingungen werden angewendet, wenn die Menge der betroffenen Zeilen nicht leer ist. Dazu gehören eindeutige Integritätsbedingungen, eindeutige Indizes, referenzielle Integritätsbedingungen, Prüfungen auf Integritätsbedingungen und Prüfungen der Klausel WITH CHECK OPTION für Sichten. Referenzielle Integritätsbedingungen mit den Löschregeln CASCADE oder SET NULL können die Aktivierung weiterer Trigger bewirken.

  Ein Verstoß gegen eine Integritätsbedingung oder die Klausel WITH CHECK OPTION führt zu einem Fehler. In diesem Fall werden alle Änderungen, die infolge der ursprünglichen Anweisung S_1 (bis dahin) ausgeführt wurden, rückgängig gemacht (Rollback).

- Verarbeiten der Nachtrigger

  Alle von S_1 aktivierten Nachtrigger (AFTER) werden in der aufsteigenden Reihenfolge ihrer Erstellung verarbeitet.

  Trigger mit der Definition FOR EACH STATEMENT verarbeiten die ausgelöste Aktion genau einmal, auch wenn die Menge der betroffenen Zeilen leer ist. Trigger mit der Definition FOR EACH ROW verarbeiten die ausgelöste Aktion einmal für jede Zeile in der Menge der betroffenen Zeilen.

  Während der Verarbeitung einer ausgelösten Aktion kann ein Fehler auftreten. In diesem Fall werden alle Änderungen, die infolge der ursprünglichen Anweisung S_1 (bis dahin) ausgeführt wurden, rückgängig gemacht (Rollback).

  Die ausgelöste Aktion eines Triggers kann ausgelöste Anweisungen enthalten, bei denen es sich um DELETE-, INSERT- oder UPDATE-Anweisungen handelt. Zum Zweck dieser Beschreibung wird eine solche Anweisung als *kaskadierende Anweisung* betrachtet.

Eine kaskadierende Anweisung ist eine DELETE-, INSERT- oder UPDATE-Anweisung, die als Teil der ausgelösten Aktion eines Nachtriggers verarbeitet wird. Eine solche Anweisung wird auf einer nachfolgenden Ebene der Triggerverarbeitung gestartet. Dies lässt sich in etwa so beschreiben, dass die ausgelöste Anweisung als neue Anweisung S_1 zugeordnet wird und alle hier beschriebenen Schritte wiederum rekursiv ausgeführt werden.

Wenn alle ausgelösten Anweisungen aus allen Nachtriggern, die von jeder Anweisung S_1 aktiviert werden, vollständig verarbeitet wurden, wird die Verarbeitung der ursprünglichen Anweisung S_1 abgeschlossen.

- R = Rollback der Änderungen auf Stand vor S_1

  Jeder Fehler (einschließlich Verstößen gegen Integritätsbedingungen) während der Verarbeitung führt zu einem Rollback aller Änderungen, die direkt oder indirekt infolge der ursprünglichen Anweisung S_1 ausgeführt wurden. Die Datenbank wird in den Zustand zurückversetzt, den sie unmittelbar vor der Ausführung der ursprünglichen Anweisung S_1 hatte.

Beispiele für das Definieren von Aktionen mit Triggern

Nehmen Sie an, ein Generalmanager möchte die Namen von Kunden, die drei oder mehr Beschwerden in den letzten 72 Stunden gesendet haben, in einer separaten Tabelle speichern. Der Generalmanager möchte außerdem informiert werden, wenn ein Kundenname in diese Tabelle mehr als einmal eingefügt wird.

Zur Definition solcher Aktionen gehen Sie zum Beispiel wie folgt vor:

- Definieren Sie eine Tabelle für unzufriedene Kunden (UNHAPPY_CUSTOMERS):

```
CREATE TABLE UNHAPPY_CUSTOMERS (
  NAME           VARCHAR (30),
  EMAIL_ADDRESS  VARCHAR (200),
  INSERTION_DATE DATE)
```

- Definieren Sie einen Trigger, der automatisch eine Zeile in UNHAPPY_CUSTOMERS einfügt, wenn 3 oder mehr Nachrichten innerhalb der letzten 3 Tage empfangen wurden (unter der Annahme, dass eine Kundentabelle CUSTOMERS vorhanden ist, die eine Spalte NAME und eine Spalte E_MAIL_ADDRESS enthält):

```
CREATE TRIGGER STORE_UNHAPPY_CUST
  AFTER INSERT ON ELECTRONIC_MAIL
  REFERENCING NEW AS N
  FOR EACH ROW
  WHEN (3 <= (SELECT COUNT(*)
              FROM ELECTRONIC_MAIL
              WHERE SENDER = N.SENDER
                AND SENDING_DATE(MESSAGE) > CURRENT DATE - 3 DAYS)
       )
  BEGIN ATOMIC
    INSERT INTO UNHAPPY_CUSTOMERS
    VALUES ((SELECT NAME
    FROM CUSTOMERS
    WHERE EMAIL_ADDRESS = N.SENDER), N.SENDER, CURRENT DATE);
  END
```

- Definieren Sie einen Trigger, der eine Nachricht an den Generalmanager sendet, wenn derselbe Kunde mehr als einmal in die Tabelle UNHAPPY_CUSTOMERS eingefügt wird (unter der Annahme, dass eine Funktion SEND_NOTE vorhanden ist, die zwei Zeichenfolgen als Eingabe akzeptiert):

```
CREATE TRIGGER INFORM_GEN_MGR
  AFTER INSERT ON UNHAPPY_CUSTOMERS
  REFERENCING NEW AS N
  FOR EACH ROW
  WHEN (1 <(SELECT COUNT(*)
```

```
            FROM UNHAPPY_CUSTOMERS
            WHERE EMAIL_ADDRESS = N.EMAIL_ADDRESS)
        )
    BEGIN ATOMIC
      VALUES(SEND_NOTE('Check customer:' CONCAT N.NAME,
                       'bigboss@vnet.ibm.com'));
    END
```

Beispiel für das Definieren von Geschäftsregeln mit Triggern

Nehmen Sie an, ein Unternehmen hat die Richtlinie, dass alle E-Mails, die sich auf Kundenbeschwerden beziehen, den Marketing-Manager Nelson in der Kopieverteilerliste zur Kenntnisnahme (CC-Liste) enthalten müssen.

Da es sich um eine Regel handelt, mag es sinnvoll erscheinen, sie als Integritätsbedingung wie im folgenden Beispiel auszudrücken (unter der Annahme, dass eine benutzerdefinierte Funktion (UDF) CC_LIST zur Überprüfung vorhanden ist):

```
ALTER TABLE ELECTRONIC_MAIL ADD
  CHECK (SUBJECT <> 'Customer complaint' OR
         CONTAINS (CC_LIST(MESSAGE), 'nelson@vnet.ibm.com') = 1)
```

Eine solche Integritätsbedingung verhindert jedoch die Einfügung von E-Mails, die sich auf Kundenbeschwerden beziehen und den Marketing-Manager nicht in der CC-Liste haben. Dies ist sicherlich im Sinne der Geschäftsregel des Unternehmens. Deren Absicht ist vielmehr, alle E-Mails an den Marketing-Manager weiterzuleiten, die sich auf Kundenbeschwerden beziehen und die nicht an den Marketing-Manager kopiert wurden. Eine solche Geschäftsregel kann nur durch einen Trigger realisiert werden, weil sie die Ausführung von Aktionen erfordert, die nicht durch deklarative Integritätsbedingungen ausgedrückt werden können. Der Trigger setzt das Vorhandensein einer Funktion SEND_NOTE voraus, die Parameter der Typen E_MAIL und 'Zeichenfolge' akzeptiert.

```
CREATE TRIGGER INFORM_MANAGER
  AFTER INSERT ON ELECTRONIC_MAIL
  REFERENCING NEW AS N
  FOR EACH ROW
  WHEN (N.SUBJECT = 'Customer complaint' AND
    CONTAINS (CC_LIST(MESSAGE), 'nelson@vnet.ibm.com') = 0)
  BEGIN ATOMIC
    VALUES(SEND_NOTE(N.MESSAGE, 'nelson@vnet.ibm.com'));
  END
```

Beispiel für das Verhindern von Operationen an Tabellen mithilfe von Triggern

Angenommen, es soll verhindert werden, dass nicht zustellbare E-Mails in einer Tabelle mit dem Namen ELECTRONIC_MAIL gespeichert werden. Zu diesem Zweck muss die Ausführung bestimmter SQL-Anweisungen INSERT vermieden werden.

Es gibt zwei Möglichkeiten, dies zu erreichen:

- Sie können einen Vortrigger (BEFORE) definieren, der einen Fehler zurückgibt, wenn der Betreff (SUBJECT) einer E-Mail die Zeichenfolge *undelivered mail* ('nicht zugestellte E-Mail') enthält:

  ```
  CREATE TRIGGER BLOCK_INSERT
    NO CASCADE BEFORE INSERT ON ELECTRONIC_MAIL
    REFERENCING NEW AS N
    FOR EACH ROW
    WHEN (SUBJECT(N.MESSAGE) = 'undelivered mail')
  ```

```
BEGIN ATOMIC
  SIGNAL SQLSTATE '85101'
    SET MESSAGE_TEXT = ('Attempt to insert undelivered mail');
   END
```

- Sie können eine Prüfung auf Integritätsbedingung definieren, die erzwingt, dass sich Werte der neuen Spalte SUBJECT von der Zeichenfolge *undelivered mail* unterscheiden:

```
ALTER TABLE ELECTRONIC_MAIL
  ADD CONSTRAINT NO_UNDELIVERED
  CHECK (SUBJECT <> 'undelivered mail')
```

Kapitel 16. Sequenzen

Eine *Sequenz* ist ein Datenbankobjekt, das die automatische Generierung von Werten wie zum Beispiel Schecknummern ermöglicht. Sequenzen eignen sich ideal für die Aufgabe der Generierung eindeutiger Schlüsselwerte. Anwendungen können Sequenzen verwenden, um mögliche Probleme in Bezug auf den gemeinsamen Zugriff und die Leistung infolge von Spaltenwerten zu vermeiden, die für die Protokollierung von numerischen Werten verwendet werden. Der Vorteil von Sequenzen im Vergleich zu außerhalb der Datenbank erstellten numerischen Werten besteht darin, dass der Datenbankserver die generierten numerischen Werte protokollieren kann. Durch einen Systemabsturz mit anschließendem Wiederanlauf werden keine doppelten Zahlenwerte generiert.

Die generierten Sequenznummern haben folgende Eigenschaften:

- Die Werte können einen beliebigen exakten numerischen Datentyp (Exact Numeric) ohne Kommastellen aufweisen. Solche Datentypen sind: SMALLINT, BIGINT, INTEGER und DECIMAL.
- Aufeinander folgende Werte können sich um ein beliebiges angegebenes ganzzahliges Inkrement unterscheiden. Der Standardwert für das Inkrement ist 1.
- Der Wert des Zählers ist wiederherstellbar. Der Wert des Zählers wird mithilfe der Protokolle wiederhergestellt, wenn eine Recovery erforderlich ist.
- Werte können in einem Cache zwischengespeichert werden, wenn dies die Leistung verbessert. Durch das Vorabzuordnen und Speichern von Werten im Cache wird die synchrone E/A für das Protokoll verringert, wenn Werte für die Sequenz generiert werden. Im Fall eines Systemfehlers werden alle zwischengespeicherten Werte, die noch nicht verwendet wurden, als verloren eingestuft. Der für CACHE angegebene Wert ist die maximale Anzahl der Sequenzwerte, die verloren gehen könnten.

Für Sequenzen können zwei Ausdrücke verwendet werden:

- **Ausdruck NEXT VALUE:** Dieser Ausdruck gibt den nächsten Wert für die angegebene Sequenz zurück. Eine neue Sequenznummer wird generiert, wenn ein Ausdruck NEXT VALUE den Namen der Sequenz angibt. Wenn es jedoch in einer Abfrage mehrere Instanzen eines Ausdrucks NEXT VALUE gibt, die denselben Sequenznamen angeben, wird der Zähler für die Sequenz nur einmal für jede Zeile des Ergebnisses erhöht und alle Instanzen des Ausdrucks NEXT VALUE geben für jede Zeile des Ergebnisses den gleichen Wert zurück.
- **Ausdruck PREVIOUS VALUE:** Dieser Ausdruck gibt den zuletzt generierten Wert für die angegebene Sequenz für eine vorherige Anweisung innerhalb des aktuellen Anwendungsprozesses zurück. Dies bedeutet, dass der Ausdruck PREVIOUS VALUE für eine Verbindung auch dann einen konstanten Wert beibehält, wenn eine andere Verbindung den Ausdruck NEXT VALUE aufruft.

Vollständige und detaillierte Informationen und Beispiele zu diesen Ausdrücken finden Sie in der „Referenz zu Sequenzen" in *SQL Reference Volume 1*.

Entwerfen von Sequenzen

Beim Entwerfen von Sequenzen müssen Sie die Unterschiede berücksichtigen, die zwischen Identitätsspalten und Sequenzen bestehen, und feststellen, welche dieser Komponenten sich für Ihre Umgebung besser eignet. Wenn Sie sich für die Verwendung von Sequenzen entscheiden, müssen Sie die verfügbaren Optionen und Parametern kennen.

Informationen zu diesem Vorgang

Lesen Sie vor dem Entwerfen von Sequenzen die Informationen unter „Sequenzen im Vergleich zu Identitätsspalten" auf Seite 534.

Sequenzen sind einfach zu definieren und zu erstellen und verfügen darüber hinaus über eine Reihe zusätzlicher Optionen, die Ihnen ein höheres Maß an Flexibilität beim Generieren der Werte bieten:

- Auswahl unter verschiedenen Datentypen (SMALLINT, INTEGER, BIGINT, DECIMAL)
- Ändern von Anfangswerten (START WITH)
- Ändern des Sequenzinkrements einschließlich der Angabe der Erhöhungs- oder Reduzierungswerte (INCREMENT BY)
- Definieren von Mindest- und Maximalwerten für den Beginn und das Ende der Sequenznummernvergabe (MINVALUE/MAXVALUE)
- Aktivieren des Umlaufs von Werten, sodass Sequenzen wieder von vorne beginnen können, oder Inaktivieren dieses Umlaufs (CYCLE/NO CYCLE)
- Aktivieren des Cachings von Sequenzwerten zur Verbesserung der Leistung oder Inaktivieren des Cachings (CACHE/NO CACHE)

Auch nach dem Generieren der Sequenz können viele dieser Werte geändert werden. Sie können beispielsweise einen anderen Anfangswert definieren, der vom Wochentag abhängt. Ein weiteres praktisches Beispiel zur Verwendung von Sequenzen stellt das Generieren und die Verarbeitung von Bankschecks dar. Die Reihenfolge von Bankschecknummern ist äußerst wichtig, und es hat ernste Konsequenzen, wenn eine Serie von Sequenznummern verloren geht oder wenn es zu Fehlern kommt.

Zur Verbesserung der Leistung sollten Sie auch mit der Funktionsweise der Option CACHE vertraut sein und diese ggf. verwenden. Mit dieser Option wird für den Datenbankmanager angegeben, wie viele Sequenzwerte vom System generiert werden sollen, bevor auf den Katalog zurückgegriffen wird, um eine weitere Gruppe von Sequenzen zu generieren. Wenn keine andere Angabe erfolgt, wird für CACHE der Standardwert 20 verwendet. Bei Verwendung des Standardwerts generiert der Datenbankmanager automatisch 20 sequenzielle Werte im Speicher (1, 2, ..., 20), wenn der erste Sequenzwert angefordert wird. Sobald eine neue Sequenznummer erforderlich ist, wird der nächste Wert aus diesem Hauptspeichercache abgerufen. Wenn die in diesem Cache gespeicherten Werte verbraucht sind, generiert der Datenbankmanager die nächsten 20 Werte (21, 22, ..., 40).

Durch die Implementierung des Sequenznummerncachings muss der Datenbankmanager nicht laufend die Katalogtabellen abfragen, um den nächsten Wert abzurufen. Hierdurch wird die zusätzliche Verarbeitungszeit für das Abrufen von Sequenznummern reduziert, es können sich jedoch möglicherweise Lücken in den Sequenzen ergeben, wenn ein Systemfehler auftritt oder wenn das System heruntergefahren wird. Wenn Sie beispielsweise für den Sequenzcache den Wert 100 defi-

nieren möchten, dann speichert der Datenbankmanager 100 Werte dieser Nummern im Cache und richtet auch den Systemkatalog so ein, dass angezeigt wird, dass die nächste Wertesequenz bei 201 beginnen soll. Wenn die Datenbank heruntergefahren wird, dann beginnt die nächste Gruppe von Sequenznummern bei 201. Die zwischen 101 und 200 generierten Nummern innerhalb der Gruppe von Sequenzen gehen verloren, wenn sie nicht verwendet wurden. Wenn das Auftreten von Lücken in den generierten Werten in Ihrer Anwendung nicht akzeptabel ist, müssen Sie für den Cachingwert die Einstellung NO CACHE definieren, obwohl dies einen höheren Systemaufwand zur Folge hat.

Weitere Informationen zu allen verfügbaren Optionen und den zugehörigen Werten finden Sie in den Informationen zur Anweisung CREATE SEQUENCE.

Verwalten des Sequenzverhaltens

Sie können das Verhalten von Sequenzen anpassen, sodass dieses optimal auf die Anforderungen Ihrer Anwendung zugeschnitten ist. Sie können die Attribute einer Sequenz ändern, indem Sie zum Erstellen einer neuen Sequenz die Anweisung CREATE SEQUENCE und für eine bereits vorhandene Sequenz die Anweisung ALTER SEQUENCE eingeben.

Im Folgenden sind verschiedene Attribute einer Sequenz aufgeführt, die angegeben werden können:

Datentyp
Die Klausel AS der Anweisung CREATE SEQUENCE gibt den numerischen Datentyp der Sequenz an. Der Datentyp legt die möglichen Mindest- und Maximalwerte der Sequenz fest. Die Mindest- und Maximalwerte für einen Datentyp sind in *SQL Reference* aufgeführt. Sie können den Datentyp einer Sequenz nicht ändern. Stattdessen müssen Sie die Sequenz löschen, indem Sie die Anweisung DROP SEQUENCE eingeben und anschließend die Anweisung CREATE SEQUENCE mit dem neuen Datentyp eingeben.

Anfangswert
Die Klausel START WITH der Anweisung CREATE SEQUENCE gibt den Anfangswert der Sequenz an. Die Klausel RESTART WITH der Anweisung ALTER SEQUENCE setzt den Wert der Sequenz auf einen angegebenen Wert zurück.

Mindestwert
Die Klausel MINVALUE definiert den Mindestwert einer Sequenz.

Maximalwert
Die Klausel MAXVALUE definiert den Maximalwert einer Sequenz.

Inkrementwert
Die Klausel INCREMENT BY definiert den Wert, den alle NEXT VALUE-Ausdrücke zum aktuellen Wert der Sequenz hinzufügen. Um den Wert der Sequenz zu verringern, müssen Sie einen negativen Wert angeben.

Sequenzzyklus
Die Klausel CYCLE bewirkt, dass der Wert einer Sequenz bei Erreichen des Maximal- oder des Mindestwertes den entsprechenden Mindest- bzw. Maximalwert generiert, wenn der Ausdruck NEXT VALUE das nächste Mal verwendet wird.

Anmerkung: Die Klausel CYCLE sollte nur dann verwendet werden, wenn keine eindeutigen Zahlenwerte benötigt werden oder wenn gewährleistet

werden kann, dass ältere Sequenzwerte nicht mehr verwendet werden, wenn die Sequenz einen Zyklus durchläuft.

Um beispielsweise eine Sequenz mit dem Namen `id_werte` zu erstellen, die mit einem Mindestwert von 0 beginnt, für die ein Maximalwert von 1000 definiert ist und die den vorhandenen Wert bei jeder Ausführung des Ausdrucks NEXT VALUE um 2 erhöht und wieder zum Mindestwert zurückkehrt, wenn der Maximalwert erreicht ist, müssen Sie folgende Anweisung eingeben:

```
CREATE SEQUENCE id_werte
  START WITH 0
  INCREMENT BY 2
  MAXVALUE 1000
  CYCLE
```

Anwendungsleistung und Sequenzen

Ähnlich wie die Verwendung von Identitätsspalten führt auch die Verwendung von Sequenzen zum Generieren von Werten gegenüber anderen Methoden im Allgemeinen zu einer Verbesserung der Anwendungsleistung. Als Alternative zu Sequenzen können Sie auch eine einspaltige Tabelle erstellen, in der der aktuelle Wert gespeichert wird, und diesen Wert entweder mithilfe eines Triggers oder unter der Steuerung der Anwendung erhöhen. In einer verteilten Umgebung, in der Anwendungen gleichzeitig auf die einspaltige Tabelle zugreifen, können die Sperren, die gesetzt werden müssen, um den seriellen Zugriff zu erzwingen, zu erheblichen Leistungseinbußen führen.

Durch Sequenzen können die Probleme vermieden werden, die sich bei Verwendung von Sperren in Bezug auf einspaltige Tabellen ergeben. Zur Verbesserung der Antwortzeiten können die Sequenzwerte im Cache zwischengespeichert werden. Um die Leistung von Anwendungen, die mit Sequenzen arbeiten, zu maximieren, sollten Sie sicherstellen, dass in Ihren Sequenzcaches eine angemessene Menge von Sequenzwerten gespeichert werden kann. Die Klausel CACHE der Anweisungen CREATE SEQUENCE und ALTER SEQUENCE gibt die maximale Anzahl von Sequenzwerten an, die vom Datenbankmanager generiert und im Speicher abgelegt werden können.

Wenn die Sequenz Werte in einer bestimmten Reihenfolge generieren muss, ohne dass diese Reihenfolge Lücken aufgrund von Systemausfällen oder einer Inaktivierung der Datenbank aufweist, müssen Sie in der Anweisung CREATE SEQUENCE die Klauseln ORDER und NO CACHE verwenden. Die Klausel NO CACHE garantiert, dass in den generierten Werten keine Lücken enthalten sind. Dies führt zu einer gewissen Reduzierung der Anwendungsleistung, da die Sequenz bei der Generierung eines neuen Werts immer einen Eintrag in das Datenbankprotokoll schreiben muss. Beachten Sie hierbei, dass Lücken trotzdem auftreten können, wenn eine Transaktion mit einem Rollback zurückgesetzt wird und den angeforderten Sequenzwert nicht tatsächlich benutzt.

Sequenzen im Vergleich zu Identitätsspalten

Obwohl Sequenzen und Identitätsspalten in DB2-Anwendungen demselben Zweck zu dienen scheinen, besteht zwischen diesen beiden Elementen ein wichtiger Unterschied. Eine Identitätsspalte generiert automatisch Werte für eine Spalte in einer einzelnen Tabelle und verwendet hierzu das Dienstprogramm **LOAD**. Eine Sequenz generiert auf Anforderung sequenzielle Werte, die in einer beliebigen SQL-Anweisung benutzt werden können. Hierzu wird die Anweisung CREATE SEQUENCE verwendet.

Identitätsspalten

Identitätsspalten ermöglichen dem Datenbankmanager das automatische Generieren eines eindeutigen numerischen Wertes für jede Zeile, die der Tabelle hinzugefügt wird. Wenn Sie eine Tabelle erstellen und jede einzelne Zeile, die der Tabelle hinzugefügt wird, eindeutig gekennzeichnet sein muss, können Sie der Tabellendefinition über die Anweisung CREATE TABLE eine Identitätsspalte hinzufügen:

```
CREATE TABLE tabellenname
  (spaltenname_1 INT,
   spaltenname_2, DOUBLE,
   spaltenname_3 INT NOT NULL GENERATED ALWAYS AS IDENTITY
     (START WITH wert_1, INCREMENT BY wert_2))
```

In diesem Beispiel gibt die dritte Spalte die Identitätsspalte an. Eines der Attribute, das Sie definieren können, ist der Wert, der in der Spalte verwendet wird, um jede Zeile beim Hinzufügen eindeutig zu definieren. Der Wert, der nach der Klausel INCREMENT BY angegeben wird, definiert, um welchen Wert nachfolgende Werte des Identitätsspalteninhalts jeweils für jede zur Tabelle hinzugefügte Zeile erhöht werden sollen.

Nach ihrer Erstellung können die Identitätseigenschaften mit der Anweisung ALTER TABLE geändert oder entfernt werden. Sie können die Anweisung ALTER TABLE auch verwenden, um Identitätseigenschaften für andere Spalten hinzuzufügen.

Sequenzen

Sequenzen ermöglichen das automatische Generieren von Werten. Sequenzen eignen sich ideal für die Aufgabe der Generierung eindeutiger Schlüsselwerte. Anwendungen können Sequenzen verwenden, um mögliche Probleme in Bezug auf den gemeinsamen Zugriff und die Leistung infolge der Generierung eines eindeutigen Zählers mit anderen Methoden zu vermeiden. Im Gegensatz zu einer Identitätsspalte ist eine Sequenz weder an eine bestimmte Tabellenspalte noch an eine eindeutige Tabellenspalte gebunden, die gleichzeitig die einzige Zugriffsmöglichkeit darstellt.

Eine Sequenz kann erstellt und zu einem späteren Zeitpunkt geändert werden, sodass sie Werte generiert, indem Werte entweder unbegrenzt erhöht oder reduziert werden oder indem ein benutzerdefinierter Grenzwert verwendet wird, nach dessen Erreichen die Verarbeitung gestoppt wird. Alternativ kann auch ein benutzerdefinierter Grenzwert verwendet werden, nach dessen Erreichen die Verarbeitung in Form eines Zyklus wieder von vorne beginnt.

Das folgende Beispiel zeigt, wie eine Sequenz mit dem Namen `orderseq` erstellt wird:

```
CREATE SEQUENCE orderseq
  START WITH 1
  INCREMENT BY 1
  NOMAXVALUE
  NOCYCLE
  CACHE 50
```

In diesem Beispiel beginnt die Sequenz bei 1 und wird unbegrenzt jeweils um den Wert 1 erhöht. Es gibt keinen Grund, zum Anfang zurückzukehren und wieder mit 1 zu beginnen, da keine obere Begrenzung zugeordnet wurde. Der Parameter CACHE gibt die maximale Anzahl von Sequenzwerten an, die vom Datenbankmanager vorab zugeordnet und im Hauptspeicher zwischengespeichert werden.

Erstellen von Sequenzen

Geben Sie zur Erstellung von Sequenzen die Anweisung CREATE SEQUENCE ein. Im Gegensatz zu einem Identitätsspaltenattribut ist eine Sequenz weder an eine bestimmte Tabellenspalte noch an eine eindeutige Tabellenspalte gebunden, die gleichzeitig die einzige Zugriffsmöglichkeit darstellt.

Informationen zu diesem Vorgang

Die Positionen, an denen Ausdrücke NEXT VALUE oder PREVIOUS VALUE verwendet werden können, unterliegen verschiedenen Einschränkungen. Eine Sequenz kann erstellt oder geändert werden, sodass mit einer der folgenden Methoden Werte generiert werden:

- Monoton erhöhen oder vermindern (d. h. um einen konstanten Betrag ändern) ohne Begrenzung
- Monoton erhöhen oder vermindern bis zu einer benutzerdefinierten Begrenzung oder einem benutzerdefinierten Stopp
- Monoton erhöhen oder vermindern bis zu einer benutzerdefinierten Begrenzung und regelmäßiges Zurücksetzen an den Anfang zum erneuten Start

Anmerkung: Beim Wiederherstellen von Datenbanken, die mit Sequenzen arbeiten, sollten Sie besonders vorsichtig vorgehen: Wenn bei Sequenzwerten, die außerhalb der Datenbank verwendet werden, z. B. Sequenznummern für Bankschecks, die Datenbank auf einen Status vor dem Auftreten des Datenbankfehlers zurückgesetzt wird, dann kann dies bei bestimmten Sequenzen zur Generierung doppelter Werte führen. Um mögliche Doppelwerte zu vermeiden, sollten Datenbanken mit Sequenzwerten, die außerhalb der Datenbank verwendet werden, nicht in einem Zustand wiederhergestellt werden, in dem sich diese zu einem früheren Zeitpunkt befunden haben.

Geben Sie zum Erstellen einer Sequenz mit dem Namen `order_seq`, für die bei allen Optionen die Standardwerte benutzt werden, in einem Anwendungsprogramm oder mithilfe der entsprechenden dynamischen SQL-Anweisungen die folgende Anweisung ein:

```
CREATE SEQUENCE order_seq
```

Diese Sequenz beginnt mit 1 und wird ohne Begrenzung immer um den Wert 1 erhöht.

Dieses Beispiel kann z. B. für die Darstellung der Verarbeitungsabläufe bei einer Reihe von Bankschecks verwendet werden, deren Nummern bei 101 beginnen und bis 200 reichen. Die erste Reihe hat Nummern zwischen 1 und 100. Die Sequenz beginnt bei 101 und wird jeweils um den Wert 1 erhöht, bis der obere Grenzwert von 200 erreicht ist. Außerdem wurde NOCYCLE angegeben, sodass keine doppelten Schecknummern erzeugt werden können. Die dem Parameter CACHE zugeordnete Anzahl gibt die maximale Anzahl der Sequenzwerte an, die der Datenbankmanager vorab zuordnet und speichert.

```
CREATE SEQUENCE order_seq
   START WITH 101
   INCREMENT BY 1
   MAXVALUE 200
   NOCYCLE
   CACHE 25
```

Weitere Informationen zu dieser und weiteren Optionen sowie zu den geltenden Berechtigungsvoraussetzungen finden Sie in den Informationen zur Anweisung CREATE SEQUENCE.

Generieren sequenzieller Werte

Das Generieren sequenzieller Werte stellt bei der Entwicklung von Datenbankanwendungen häufig ein Problem dar. Die beste Lösung dieses Problems besteht in der Verwendung von Sequenzen und Sequenzausdrücken in SQL. Jede *Sequenz* stellt ein eindeutig benanntes Datenbankobjekt dar, auf das ausschließlich über Sequenzausdrücke zugegriffen werden kann.

Es gibt zwei *Sequenzausdrücke*: den Ausdruck PREVIOUS VALUE und den Ausdruck NEXT VALUE. Der Ausdruck PREVIOUS VALUE gibt den Wert zurück, der für die angegebene Sequenz im Anwendungsprozess zuletzt generiert wurde. NEXT VALUE-Ausdrücke, die in derselben Anweisung wie der Ausdruck PREVIOUS VALUE aufgeführt sind, wirken sich nicht auf den vom Ausdruck PREVIOUS VALUE in dieser Anweisung generierten Wert aus. Der Sequenzausdruck NEXT VALUE erhöht den Wert der Sequenz und gibt den neuen Wert der Sequenz zurück.

Geben Sie zur Erstellung einer Sequenz die Anweisung CREATE SEQUENCE ein. Um beispielsweise eine Sequenz mit dem Namen `id_werte` zu erstellen und hierbei die Standardattribute zu verwenden, müssen Sie die folgende Anweisung eingeben:

```
CREATE SEQUENCE id_werte
```

Geben Sie zum Generieren des ersten Werts in der Anwendungssitzung für die Sequenz eine Anweisung VALUES mit dem Ausdruck NEXT VALUE ein:

```
VALUES NEXT VALUE FOR id_werte

1
-----------
          1

            1 Satz/Sätze ausgewählt.
```

Um den Wert einer Spalte mit dem nächsten Wert der Sequenz zu aktualisieren, müssen Sie den Ausdruck NEXT VALUE wie folgt in die Anweisung UPDATE einfügen:

```
UPDATE staff
  SET id = NEXT VALUE FOR id_werte
  WHERE id = 350
```

Um eine neue Zeile in eine Tabelle einzufügen und hierbei den nächsten Wert der Sequenz zu verwenden, müssen Sie den Ausdruck NEXT VALUE wie folgt in die Anweisung INSERT einfügen:

```
INSERT INTO staff (id, name, dept, job)
  VALUES (NEXT VALUE FOR id_werte, 'Kandil', 51, 'Mgr')
```

Ermitteln der Bedingungen zur Verwendung von Identitätsspalten oder Sequenzen

Zwischen Identitätsspalten und Sequenzen gibt es sowohl Gemeinsamkeiten als auch Unterschiede. Die Merkmale von Identitätsspalten und Sequenzen können bei der Analyse und beim Entwurf von Datenbanken und Anwendungen verwendet werden.

Abhängig vom Design Ihrer Datenbank und der Anwendungen, die diese Datenbank nutzen, können Sie anhand der folgenden Merkmale feststellen, wann Identitätsspalten und wann Sequenzen verwendet werden sollten.

Merkmale von Identitätsspalten

- Eine Identitätsspalte generiert automatisch Werte für eine einzelne Tabelle.
- Wenn eine Identitätsspalte als GENERATED ALWAYS definiert ist, werden die verwendeten Werte immer vom Datenbankmanager generiert. Es ist nicht zulässig, dass Anwendungen bei der Änderung des Inhalts der Tabelle eigene Werte bereitstellen.
- Nach dem Einfügen einer Zeile kann der generierte Identitätswert entweder mit der Funktion IDENTITY_VAL_LOCAL() oder durch Rückauswahl der Identitätsspalte aus der Einfügeoperation mit Hilfe der Anweisung SELECT FROM INSERT abgerufen werden.
- Zum Generieren von Identitätswerten (IDENTITY-Werten) kann das Dienstprogramm LOAD eingesetzt werden.

Merkmale von Sequenzen

- Sequenzen sind an keine bestimmte Tabelle gebunden.
- Sequenzen generieren sequenzielle Werte, die in einer beliebigen SQL- oder XQuery-Anweisung verwendet werden können.

  Da eine Sequenz von jeder Anwendung verwendet werden kann, gibt es zwei Ausdrücke, die das Abrufen des nächsten Wertes in der angegebenen Sequenz und des Wertes, der vor der gerade ausgeführten Anweisung generiert wurde, steuern. Der Ausdruck PREVIOUS VALUE gibt den zuletzt generierten Wert für die angegebene Sequenz für eine vorherige Anweisung innerhalb der aktuellen Sitzung zurück. Der Ausdruck NEXT VALUE gibt den nächsten Wert für die angegebene Sequenz zurück. Die Verwendung dieser Ausdrücke ermöglicht es, denselben Wert in verschiedenen SQL- und XQuery-Anweisungen innerhalb mehrerer Tabellen zu verwenden.

Ändern von Sequenzen

Ändern Sie die Attribute einer vorhandenen Sequenz mit der Anweisung ALTER SEQUENCE.

Informationen zu diesem Vorgang

Die folgenden Attribute der Sequenz können geändert werden:

- Ändern des Inkrements zwischen künftigen Werten
- Einrichten neuer Minimal- oder Maximalwerte
- Ändern der Anzahl zwischengespeicherter Sequenznummern
- Ändern der Anweisung, ob die Sequenz einen Zyklus ausführen soll oder nicht
- Ändern der Anweisung, ob die Sequenznummern in der Anforderungsreihenfolge generiert werden müssen
- Neustarten der Sequenz

Es gibt zwei Aufgaben, die nicht Teil der Erstellung der Sequenz sind. Dies sind:

- **RESTART:** Setzt die Sequenz auf den bei der Erstellung implizit oder explizit als Anfangswert angegebenen Wert zurück.

- **RESTART WITH *numerische-konstante*:** Setzt die Sequenz auf den exakten Wert der numerischen Konstanten zurück. Die numerische Konstante kann ein beliebiger positiver oder negativer Wert, der von einem etwaigen Dezimalzeichen keine anderen Ziffern als Nullen aufweisen darf.

Einschränkungen

Der Datentyp einer Sequenz kann nicht geändert werden. Sie müssen stattdessen die aktuelle Sequenz löschen und anschließend eine Sequenz unter Angabe des neuen Datentyps erstellen.

Ergebnisse

Nach dem Neustarten einer Sequenz oder dem Wechsel zu CYCLE können doppelte Sequenznummern auftreten. Von der Anweisung ALTER SEQUENCE sind nur zukünftige Sequenznummern betroffen.

Alle im Cache gespeicherten Sequenzwerte, die vom Datenbankmanager nicht verwendet werden, gehen beim Ändern einer Sequenz verloren.

Anzeigen von Sequenzdefinitionen

Zum Anzeigen der Verweisinformationen, die einer Sequenz zugeordnet sind, oder zum Anzeigen der Sequenz selbst können Sie die Anweisung VALUES mit der Option PREVIOUS VALUE benutzen.

Vorgehensweise

Zum Anzeigen des aktuellen Werts der Sequenz müssen Sie eine Anweisung VALUES mit dem Ausdruck PREVIOUS VALUE eingeben:

```
VALUES PREVIOUS VALUE FOR id_werte

1
-----------
          1

            1 Satz/Sätze ausgewählt.
```

Sie können den aktuellen Wert der Sequenz wiederholt abrufen. Der von der Sequenz zurückgegebene Wert ändert sich erst, wenn Sie den Ausdruck NEXT VALUE eingeben. Dies gilt auch dann, wenn eine andere Verbindung zur gleichen Zeit Sequenzwerte liest.

Beispiel

Im folgenden Beispiel gibt der Ausdruck PREVIOUS VALUE den Wert 1 zurück, bis mit dem Ausdruck NEXT VALUE in der aktuellen Verbindung der Wert der Sequenz erhöht wird:

```
VALUES PREVIOUS VALUE FOR id_werte

1
-----------
          1

            1 Satz/Sätze ausgewählt.
VALUES PREVIOUS VALUE FOR id_werte

1
```

```
-----------
          1

            1 Satz/Sätze ausgewählt.

VALUES NEXT VALUE FOR id_werte

1
-----------
          2

            1 Satz/Sätze ausgewählt.

VALUES PREVIOUS VALUE FOR id_werte

1
-----------
          2

            1 Satz/Sätze ausgewählt.
```

Löschen von Sequenzen

Verwenden Sie zum Löschen einer Sequenz die Anweisung DROP.

Vorbereitende Schritte

Beim Löschen von Sequenzen muss die Berechtigungs-ID der Anweisung über die Berechtigung DBADM verfügen.

Einschränkungen

Vom System erstellte Sequenzen für Identitätsspalten können nicht mithilfe der Anweisung DROP SEQUENCE gelöscht werden.

Vorgehensweise

Geben Sie Folgendes ein, um eine bestimmte Sequenz zu löschen:

```
DROP SEQUENCE sequenzname
```

Dabei ist *sequenzname* der Name der zu löschenden Sequenz, der den impliziten bzw. expliziten Schemanamen enthält, um eine vorhandene Sequenz genau anzugeben.

Ergebnisse

Wenn eine Sequenz gelöscht wird, werden alle Zugriffsrechte für die Sequenz ebenfalls gelöscht.

Beispiele zur Codierung von Sequenzen

Zahlreiche Anwendungen, die geschrieben werden, benötigen eine Sequenznummer, mit der Rechnungsnummern, Kundennummern und weitere Objekte protokolliert werden können, die um den Wert eins erhöht werden, sobald ein neues Element erforderlich ist. Der Datenbankmanager kann Werte in einer Tabelle mithilfe von Identitätsspalten automatisch erhöhen. Obwohl dieses Verfahren bei einzelnen Tabellen zufriedenstellend funktioniert, stellt es nicht die komfortabelste Möglichkeit zum Generieren eindeutiger Werte dar, die in mehreren Tabellen verwendet werden müssen.

Das *Sequenzobjekt* ermöglicht Ihnen die Erstellung eines Wertes, der unter Programmiersteuerung erhöht wird und in zahlreichen Tabellen verwendet werden kann. Das folgende Beispiel zeigt eine Sequenznummer, die für Kundennummern erstellt wird und für die der Datentyp Integer verwendet wird:

```
CREATE SEQUENCE kundennummer AS INTEGER
```

Standardmäßig beginnt die Sequenznummer bei eins und erhöht sich jeweils um den Wert eins. Sie weist den Datentyp INTEGER auf. Die Anwendung ruft mit der Funktion NEXT VALUE den nächsten Wert innerhalb der Sequenz ab. Diese Funktion generiert den nächsten Wert für die Sequenz, der dann für nachfolgende SQL-Anweisungen verwendet werden kann:

```
VALUES NEXT VALUE FOR kundennummer
```

Anstatt mit der Funktion VALUES die nächste Nummer zu generieren, kann der Programmierer diese Funktion auch in einer Anweisung INSERT verwenden. Wenn beispielsweise die erste Spalte der Kundentabelle die Kundennummer enthält, dann kann die folgende Anweisung INSERT definiert werden:

```
INSERT INTO kunden VALUES
  (NEXT VALUE FOR kundennummer, 'kommentar', ...)
```

Wenn die Sequenznummer für Einfügungen in andere Tabellen verwendet werden muss, dann kann die Funktion PREVIOUS VALUE benutzt werden, um den zuvor generierten Wert abzurufen. Wenn beispielsweise die soeben erstellte Kundennummer für einen nachfolgenden Rechnungsdatensatz verwendet werden soll, dann umfasst der SQL-Code die Funktion PREVIOUS VALUE:

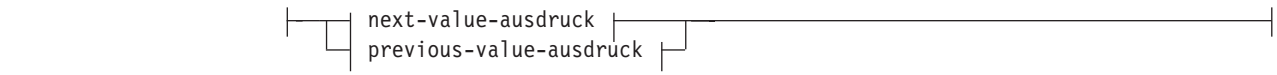
```
INSERT INTO rechnungen
  (34,PREVIOUS VALUE FOR kundennummer, 234.44, ...)
```

Die Funktion PREVIOUS VALUE kann innerhalb der Anwendung mehrfach verwendet werden und gibt nur den Wert zurück, der von dieser Anwendung zuletzt generiert wurde. Möglicherweise haben nachfolgende Transaktionen die Sequenz bereits auf einen anderen Wert erhöht, Sie erhalten jedoch immer die zuletzt generierte Zahl.

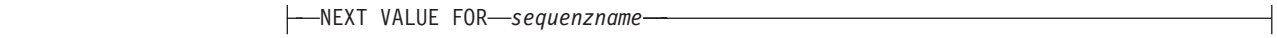
Sequenzverweis

Ein Sequenzverweis ist ein Ausdruck, der auf eine Sequenz verweist, die beim Anwendungsserver definiert ist.

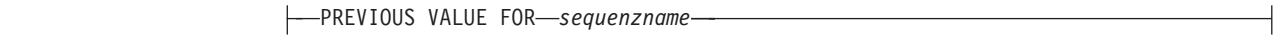
sequenzverweis:



next-value-ausdruck:



previous-value-ausdruck:



NEXT VALUE FOR *sequenzname*
Ein Ausdruck NEXT VALUE generiert den nächsten Wert für die durch *sequenzname* angegebene Sequenz und gibt ihn zurück.

PREVIOUS VALUE FOR *sequenzname*
Ein Ausdruck PREVIOUS VALUE gibt den zuletzt generierten Wert für die angegebene Sequenz für eine vorherige Anweisung innerhalb des aktuellen Anwendungsprozesses zurück. Auf diesen Wert kann wiederholt verwiesen werden, indem Ausdrücke PREVIOUS VALUE verwendet werden, die den Namen der Sequenz angeben. Es können mehrere Instanzen von Ausdrücken PREVIOUS VALUE, die denselben Sequenznamen angeben, in einer einzigen Anweisung enthalten sein: Sie alle geben denselben Wert zurück. In einer Umgebung mit partitionierten Datenbanken ist es möglich, dass ein Ausdruck PREVIOUS VALUE nicht den zuletzt generierten Wert zurückgibt.

Ein Ausdruck PREVIOUS VALUE kann nur verwendet werden, wenn im aktuellen Anwendungsprozess entweder in der aktuellen oder in einer vorherigen Transaktion ein Ausdruck NEXT VALUE mit demselben Sequenznamen verwendet wurde (SQLSTATE-Wert 51035).

Hinweise

- ***Berechtigung:*** Wenn der Sequenzverweis in einer Anweisung verwendet wird, müssen die Zugriffsrechte der Berechtigungs-ID der Anweisung mindestens eine der folgenden Berechtigungen und eines der folgenden Zugriffsrechte beinhalten:
  - Berechtigung USAGE für die Sequenz
  - Berechtigung DATAACCESS
- Für eine Sequenz wird ein neuer Wert generiert, wenn ein Ausdruck NEXT VALUE den Namen dieser Sequenz angibt. Wenn es jedoch in einer Abfrage mehrere Instanzen eines Ausdrucks NEXT VALUE gibt, die denselben Sequenznamen angeben, wird der Zähler für die Sequenz nur einmal für jede Zeile des Ergebnisses erhöht und alle Instanzen des Ausdrucks NEXT VALUE geben für eine Zeile des Ergebnisses den gleichen Wert zurück.
- Derselbe Sequenzwert kann als eindeutiger Schlüsselwert in zwei separaten Tabellen verwendet werden, indem der Sequenzwert durch einen Ausdruck NEXT VALUE für die erste Zeile (dadurch wird der Sequenzwert generiert) und durch einen Ausdruck PREVIOUS VALUE für die anderen Zeilen (die Instanz von PREVIOUS VALUE bezieht sich auf den zuletzt in der aktuellen Sitzung generierten Sequenzwert) angegeben wird, wie im folgenden Beispiel gezeigt wird:

```
INSERT INTO order(orderno, cutno)
  VALUES (NEXT VALUE FOR order_seq, 123456);

INSERT INTO line_item (orderno, partno, quantity)
  VALUES (PREVIOUS VALUE FOR order_seq, 987654, 1);
```

- Ausdrücke mit NEXT VALUE und PREVIOUS VALUE können an folgenden Stellen angegeben werden:
  - Anweisung SELECT oder SELECT INTO (innerhalb der Klausel SELECT, sofern die Anweisung kein Schlüsselwort DISTINCT, keine Klausel GROUP BY oder ORDER BY und keines der Schlüsselwörter UNION, INTERSECT oder EXCEPT enthält)
  - Anweisung INSERT (in einer Klausel VALUES)
  - Anweisung INSERT (in der Klausel SELECT des Fullselects)

  - Anweisung UPDATE (in der Klausel SET, entweder gesuchte oder positionierte Anweisung UPDATE, mit der Ausnahme, dass der Ausdruck NEXT VALUE nicht in der Klausel SELECT des Fullselects eines Ausdrucks in der Klausel SET angegeben werden kann)
  - Anweisung SET variable (außer in der Klausel SELECT des Fullselects eines Ausdrucks; ein Ausdruck NEXT VALUE kann in einem Trigger angegeben werden, ein Ausdruck PREVIOUS VALUE hingegen nicht)
  - Anweisung VALUES INTO (in der Klausel SELECT des Fullselects eines Ausdrucks)
  - Anweisung CREATE PROCEDURE (im Routinenhauptteil einer SQL-Prozedur)
  - Anweisung CREATE TRIGGER in der ausgelösten Aktion (es kann ein Ausdruck NEXT VALUE angegeben werden, ein Ausdruck PREVIOUS VALUE hingegen nicht)
- Ausdrücke mit NEXT VALUE und PREVIOUS VALUE können an folgenden Stellen nicht angegeben werden (SQLSTATE-Wert 428F9):
  - Joinbedingung eines vollständigen Outer Joins
  - DEFAULT-Wert für eine Spalte in einer Anweisung CREATE oder ALTER TABLE
  - Definition für generierte Spalte in einer Anweisung CREATE oder ALTER TABLE
  - Definition für Übersichtstabelle in einer Anweisung CREATE TABLE oder ALTER TABLE
  - Bedingung einer Prüfung auf Integritätsbedingung (CHECK)
  - Anweisung CREATE TRIGGER (ein Ausdruck NEXT VALUE kann angegeben werden, jedoch kein Ausdruck PREVIOUS VALUE)
  - Anweisung CREATE VIEW
  - Anweisung CREATE METHOD
  - Anweisung CREATE FUNCTION
  - Argumentliste eines XMLQUERY-, XMLEXISTS- oder XMLTABLE-Ausdrucks
- Darüber hinaus kann ein Ausdruck NEXT VALUE nicht an folgenden Stellen angegeben werden (SQLSTATE-Wert 428F9):
  - CASE-Ausdruck
  - Parameterliste einer Spaltenfunktion
  - Unterabfrage in einem nicht explizit zulässigen Kontext (wie oben beschrieben)
  - Anweisung SELECT, für die das äußere SELECT einen Operator DISTINCT enthält
  - Joinbedingung eines Joins
  - Anweisung SELECT, für die das äußere SELECT eine Klausel GROUP BY enthält
  - Anweisung SELECT, für die das äußere SELECT mit einer anderen Anweisung SELECT durch den Gruppenoperator UNION, INTERSECT oder EXCEPT kombiniert wird
  - Verschachtelter Tabellenausdruck
  - Parameterliste einer Tabellenfunktion
  - Klausel WHERE der äußersten Anweisung SELECT oder einer Anweisung DELETE bzw. UPDATE
  - Klausel ORDER BY der äußersten Anweisung SELECT

  - Klausel SELECT des Fullselects eines Ausdrucks, in der Klausel SET einer Anweisung UPDATE
  - Anweisung IF, WHILE, DO ... UNTIL oder CASE in einer SQL-Routine
- Wenn ein Wert für eine Sequenz generiert wird, ist dieser Wert verbraucht. Bei der nächsten Anforderung eines Werts wird ein neuer Wert generiert. Dies gilt auch für den Fall, dass die Anweisung, die den Ausdruck NEXT VALUE enthält, fehlschlägt oder durch Rollback rückgängig gemacht wird.

  Wenn eine Anweisung INSERT einen Ausdruck NEXT VALUE in der VALUES-Liste für die Spalte enthält und während der Ausführung dieser Anweisung INSERT ein Fehler auftritt (sei es Fehler beim Generieren des nächsten Sequenzwerts oder ein Fehler bei einem Wert für eine andere Spalte), wird ein Einfügefehler (SQLSTATE-Wert 23505) zurückgegeben, und der generierte Wert für die Sequenz wird als verbraucht betrachtet. In einigen Fällen kann ein erneutes Absetzen derselben Anweisung INSERT zum Erfolg führen.

  Betrachten Sie zum Beispiel einen Fehler, der infolge eines vorhandenen eindeutigen Index für die Spalte auftritt, für die ein Ausdruck NEXT VALUE verwendet wurde, weil der generierte Sequenzwert bereits im Index vorhanden ist. Es ist möglich, dass der nächste für die Sequenz generierte Wert ein Wert ist, der nicht im Index vorhanden ist, sodass die nachfolgende INSERT-Operation erfolgreich ausgeführt würde.
- ***Gültigkeitsbereich von PREVIOUS VALUE:*** Der Wert von PREVIOUS VALUE bleibt bestehen, bis der nächste Wert für die Sequenz in der aktuellen Sitzung generiert, die Sequenz gelöscht oder geändert oder die Anwendungssitzung beendet wird. Der Wert wird von COMMIT- oder ROLLBACK-Anweisungen nicht berührt. Der Wert von PREVIOUS VALUE kann nicht direkt festgelegt werden, sondern ist das Ergebnis der Ausführung des Ausdrucks NEXT VALUE für die Sequenz.

  Ein gängiges Verfahren insbesondere im Hinblick auf die Leistung besteht darin, dass eine Anwendung oder ein Produkt eine Gruppe von Verbindungen verwaltet und Transaktionen an eine beliebige Verbindung weiterleitet. In solchen Fällen sollte die Verfügbarkeit des Werts von PREVIOUS VALUE für eine Sequenz nur bis zum Ende der Transaktion als sichergestellt angenommen werden. Beispiele für Fälle, in denen diese Art von Situation auftreten kann, sind Anwendungen, die mit XA-Protokollen arbeiten, ein Verbindungspooling verwenden und den Verbindungskonzentrator verwenden sowie Anwendungen, die HADR (High Availability Disaster Recovery) zur Realisierung einer Funktionsübernahme nutzen.
- Wenn beim Generieren eines Werts für eine Sequenz der Maximalwert für die Sequenz überschritten wird (bzw. bei einer absteigenden Sequenz der Minimalwert unterschritten wird) und Zyklen nicht zulässig sind, tritt ein Fehler auf (SQLSTATE-Wert 23522). In diesem Fall könnte der Benutzer die Sequenz ändern (ALTER), um den Bereich der akzeptablen Werte zu erweitern. Alternativ könnte er Zyklen für die Sequenz zulassen oder die Sequenz löschen (DROP) und eine neue Sequenz mit einem anderen Datentyp erstellen (CREATE), der einen größeren Bereich von Werten besitzt.

  Eine Sequenz kann zum Beispiel mit dem Datentyp SMALLINT definiert worden sein, sodass sie schließlich keine zuweisbaren Werte mehr generieren kann. Eine solche Sequenz können Sie löschen und erneut erstellen, um sie mit dem Datentyp INTEGER zu definieren.
- Ein Verweis auf einen Ausdruck NEXT VALUE in der Anweisung SELECT eines Cursors bezieht sich auf einen Wert, der für eine Zeile der Ergebnistabelle generiert wird. Für einen Ausdruck NEXT VALUE wird für jede Zeile, die aus der Datenbank abgerufen wird, ein Sequenzwert generiert. Wenn auf dem Client eine Zeilenblockung stattfindet, können Werte auf dem Server vor der Verarbei-

tung der Anweisung FETCH generiert worden sein. Dies kann geschehen, wenn eine Blockung der Zeilen der Ergebnistabelle stattfindet. Wenn die Clientanwendung nicht explizit alle Zeilen abruft (FETCH), die in der Datenbank gespeichert sind, werden für die Anwendung nicht die Ergebnisse aller generierten Sequenzwerte sichtbar (nämlich die für die gespeicherten Zeilen nicht, die nicht zurückgegeben wurden).

- Ein Verweis auf einen Ausdruck PREVIOUS VALUE in der Anweisung SELECT eines Cursors bezieht sich auf einen Wert, der für die angegebene Sequenz vor dem Öffnen des Cursors generiert wurde. Das Schließen des Cursors kann sich jedoch auf die Werte auswirken, die durch PREVIOUS VALUE für die angegebene Sequenz in nachfolgenden Anweisungen oder sogar, falls der Cursor erneut geöffnet wird, in derselben Anweisung zurückgegeben werden. Dies wäre zum Beispiel der Fall, wenn die Anweisung SELECT des Cursors einen Verweis auf einen Ausdruck NEXT VALUE für denselben Sequenznamen enthielte.
- ***Syntaxalternativen***: Die nachfolgend aufgeführten Syntaxalternativen werden aus Gründen der Kompatibilität mit früheren DB2-Versionen und mit anderen Datenbankprodukten unterstützt. Diese Alternativen weichen vom Standard ab und sollten nicht verwendet werden.
  - Anstelle von NEXT VALUE und PREVIOUS VALUE können auch NEXTVAL und PREVVAL angegeben werden.
  - Anstelle von NEXT VALUE FOR *sequenzname* kann auch *sequenzname*.NEXTVAL angegeben werden.
  - Anstelle von PREVIOUS VALUE FOR *sequenzname* kann auch *sequenzname*.CURRVAL angegeben werden.

Beispiele

Nehmen Sie an, es gibt eine Tabelle mit dem Namen "order" und eine Sequenz mit dem Namen "order_seq" wird folgendermaßen erstellt:

```
CREATE SEQUENCE order_seq
  START WITH 1
  INCREMENT BY 1
  NO MAXVALUE
  NO CYCLE
  CACHE 24
```

Die folgenden Beispiele zeigen, wie eine Folgenummer mit der Sequenz "order_seq" durch einen Ausdruck NEXT VALUE erstellt werden kann:

```
INSERT INTO order(orderno, custno)
  VALUES (NEXT VALUE FOR order_seq, 123456);
```

oder

```
UPDATE order
  SET orderno = NEXT VALUE FOR order_seq
  WHERE custno = 123456;
```

oder

```
VALUES NEXT VALUE FOR order_seq INTO :hv_seq;
```

Kapitel 17. Sichten

Eine *Sicht* bietet eine effektive Methode zur Darstellung von Daten, ohne sie pflegen zu müssen. Eine Sicht ist keine wirkliche Tabelle und erfordert keine permanente Speicherung. Es wird eine „virtuelle Tabelle" erstellt und verwendet.

Eine *Sicht* bietet eine alternative Möglichkeit zur Darstellung der Daten in einer oder auch in mehreren Tabellen. Sie stellt eine benannte Spezifikation einer Ergebnistabelle dar. Die Spezifikation ist eine Anweisung SELECT, die immer dann ausgeführt wird, wenn in einer SQL-Anweisung auf die Sicht Bezug genommen wird. Eine Sicht verfügt ebenso wie eine Tabelle über Spalten und Zeilen. Alle Sichten können ebenso wie Tabellen zum Abrufen von Daten verwendet werden. Ob eine Sicht in einer Einfüge-, Aktualisierungs- oder Löschoperation verwendet werden kann, hängt von ihrer Definition ab.

Eine Sicht kann alle oder einige der Spalten oder Zeilen der Tabelle enthalten, auf der sie basiert. Zum Beispiel können Sie eine Tabelle mit Abteilungsdaten (DEPARTMENT) und eine Tabelle mit Mitarbeiterdaten (EMPLOYEE) in einer Sicht verknüpfen, sodass Sie alle Mitarbeiter in einer bestimmten Abteilung auflisten können.

Abb. 50 zeigt die Beziehung zwischen Tabellen und Sichten.

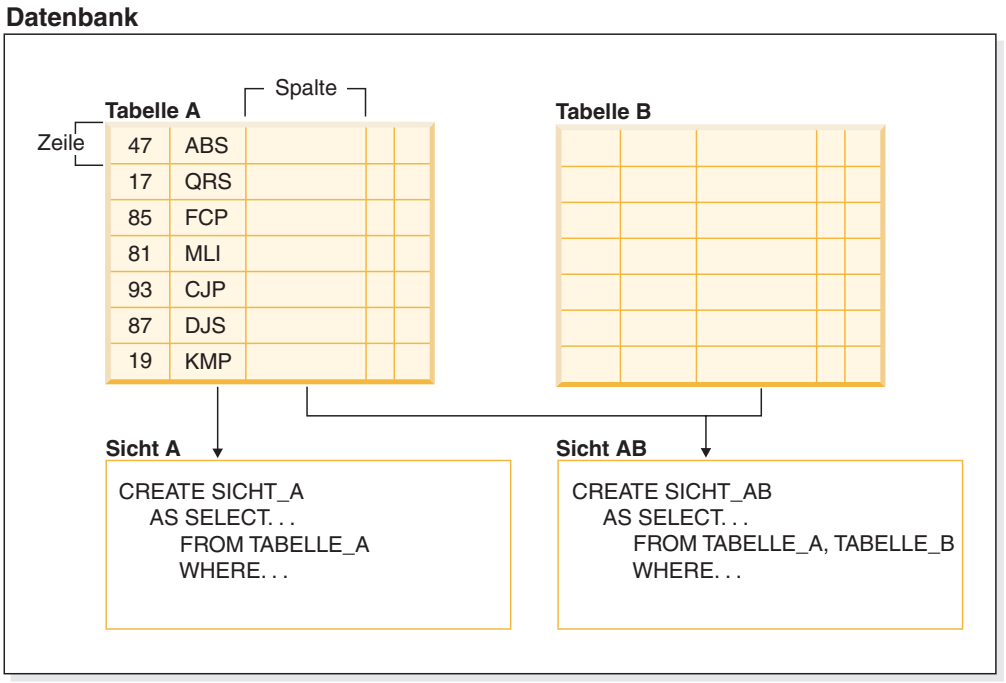


Abbildung 50. Beziehung zwischen Tabellen und Sichten

Sie können Sichten verwenden, um den Zugriff auf sensible Daten zu steuern, da mit ihnen ein bestimmter Datenbestand für unterschiedliche Benutzer in verschiedenen Darstellungen angezeigt werden kann. Beispiel: Mehrere Benutzer greifen auf eine Tabelle mit Mitarbeiterdaten zu. Ein Manager kann dann die Daten zu seinen Mitarbeitern, jedoch nicht die Daten der Mitarbeiter einer anderen Abteilung anzeigen. Ein Personalmitarbeiter kann die Einstellungstermine aller Mitarbeiter

anzeigen, jedoch nicht ihre Gehälter, ein Buchhaltungsmitarbeiter hingegen die Gehälter, jedoch nicht die Einstellungstermine. Jeder dieser Benutzer arbeitet mit einer Sicht, die von der Tabelle abgeleitet wurde. Jede Sicht erscheint für den Benutzer wie eine eigenständige Tabelle und verfügt über einen eigenen Namen.

Wenn die Spalte einer Sicht direkt aus der Spalte einer Basistabelle abgeleitet wird, dann übernimmt diese Sichtspalte alle Integritätsbedingungen, die auch für die entsprechende Spalte der Tabelle gelten. Wenn eine Sicht beispielsweise einen Fremdschlüssel der zugehörigen Tabelle enthält, dann gelten für Einfüge- und Aktualisierungsoperationen in dieser Sicht die gleichen referenziellen Integritätsbedingungen wie für die Tabelle. Wenn es sich bei der Tabelle einer Sicht um eine übergeordnete Tabelle handelt, dann gelten für Lösch- und Aktualisierungsoperationen in dieser Sicht ebenfalls die gleichen Regeln wie für Lösch- und Aktualisierungsoperationen in der Tabelle.

Eine Sicht kann den Datentyp aller Spalten aus der Ergebnistabelle ableiten oder die Typzuordnung auf der Basis der Attribute vornehmen, die für einen benutzerdefinierten strukturierten Typ gelten. Derartige Sichten werden als *typisierte Sicht* bezeichnet. Ähnlich wie bei einer typisierten Tabelle kann eine typisierte Sicht Teil einer Sichtenhierarchie sein. Eine *untergeordnete Sicht* übernimmt die Spalten der zugehörigen *übergeordneten Sicht*. Der Begriff *untergeordnete Sicht* bezieht sich auf eine typisierte Sicht und auf alle typisierten Sichten, die sich innerhalb der Sichtenhierarchie unter dieser befinden. Eine *korrekte untergeordnete Sicht* der Sicht V ist eine Sicht, die sich innerhalb der Hierarchie typisierter Sichten unterhalb der Sicht V befindet.

Eine Sicht kann funktionsunfähig werden. (Dies ist z. B. dann der Fall, wenn die zugehörige Tabelle gelöscht wird.) In einer derartigen Situation steht die Sicht nicht mehr für SQL-Operationen zur Verfügung.

Entwerfen von Sichten

Eine *Sicht* bietet eine alternative Möglichkeit zur Darstellung der Daten in einer oder auch in mehreren Tabellen. Sie stellt eine benannte Spezifikation einer Ergebnistabelle dar.

Die Spezifikation ist eine Anweisung SELECT, die immer dann ausgeführt wird, wenn in einer SQL-Anweisung auf die Sicht Bezug genommen wird. Eine Sicht verfügt ebenso wie eine Basistabelle über Spalten und Zeilen. Alle Sichten können ebenso wie die Tabellen zum Abrufen von Daten verwendet werden. Ob eine Sicht in einer Einfüge-, Aktualisierungs- oder Löschoperation verwendet werden kann, hängt von ihrer Definition ab.

Sichten werden abhängig von den zulässigen Operationen in verschiedene Klassen eingeteilt. Diese lauten wie folgt:

- Löschfähige Sichten
- Aktualisierungsfähige Sichten
- Einfügefähige Sichten
- Schreibgeschützte Sichten

Der Typ einer Sicht wird anhand ihrer Aktualisierungsmöglichkeiten festgelegt. Die Klassifizierung gibt die Art der SQL-Operation an, die für die Sicht ausgeführt werden darf.

Referenzielle Integritätsbedingungen und Prüfbedingungen werden unabhängig voneinander behandelt. Sie haben keine Auswirkungen auf die Klassifizierung der Sicht.

Beispielsweise ist es möglich, dass eine Zeile aufgrund einer referenziellen Integritätsbedingung nicht in eine Tabelle eingefügt werden kann. Wenn Sie eine Sicht auf der Basis dieser Tabelle erstellen, ist es auch nicht möglich, diese Zeile über die Sicht einzufügen. Wenn die Sicht jedoch alle Regeln für eine einfügefähige Sicht erfüllt, dann wird sie dennoch als einfügefähige Sicht betrachtet. Dies ist darauf zurückzuführen, dass die Einfügeeinschränkung für die Tabelle und nicht für die Sichtdefinition gilt.

Weitere Informationen zu diesem Thema finden Sie in den Informationen zur Anweisung CREATE VIEW.

Systemkatalogsichten

Der Datenbankmanager verwaltet eine Gruppe von Tabellen und Sichten, die Informationen zu den Daten enthalten, die seiner Steuerung unterliegen. Diese Tabellen und Sichten werden zusammen als *Systemkatalog* bezeichnet.

Der Systemkatalog enthält Informationen zur logischen und physischen Struktur von Datenbankobjekten wie z. B. Tabellen, Sichten, Indizes, Paketen und Funktionen. Darüber hinaus enthält er statistische Informationen. Der Datenbankmanager stellt sicher, dass die Beschreibungen im Systemkatalog immer korrekt sind.

Die Systemkatalogsichten funktionieren wie alle anderen Datenbanksichten. Die in ihnen gespeicherten Daten können mithilfe von SQL-Anweisungen abgefragt werden. Eine Gruppe aktualisierbarer Systemkatalogsichten kann verwendet werden, um bestimmte Werte im Systemkatalog zu ändern.

Sichten mit Prüfoption

Eine Sicht, für die WITH CHECK OPTION definiert wurde, erzwingt die Verarbeitung aller Zeilen, die mit der Anweisung SELECT für diese Sicht geändert oder eingefügt wurden. Sichten mit der Prüfoption werden auch als *symmetrische Sichten* bezeichnet. Eine symmetrische Sicht, die ausschließlich die Mitarbeiter in Abteilung 10 zurückgibt, erlaubt z. B. keine Einfügeoperationen mit Mitarbeitern anderer Abteilungen. Diese Option stellt aus diesem Grund die Integrität der in der Datenbank geänderten Daten sicher und gibt einen Fehler zurück, wenn die geltende Bedingung während einer INSERT- oder UPDATE-Operation nicht beachtet wird.

Wenn in Ihrer Anwendung die erforderlichen Regeln nicht in Form einer Prüfung auf Integritätsbedingungen in Tabellen definiert werden können oder die Regeln nicht für alle Verwendungsvorkommen der Daten gelten, dann gibt es eine alternative Vorgehensweise zur Integration der Regeln in die Anwendungslogik. Sie können eine Sicht der Tabelle erstellen, für die die Bedingungen, die für die Daten gelten, als Teil der angegebenen Klauseln WHERE und WITH CHECK OPTION definiert werden. Diese Sichtdefinition schränkt den Abruf der Daten auf die Gruppe ein, die für Ihre Anwendung zulässig ist. Darüber hinaus schränkt die Klausel WITH CHECK OPTION bei Sichten, die aktualisiert werden können, Aktualisierungs-, Einfüge- und Löschoperationen in den Zeilen ein, die für Ihre Anwendung gelten.

Die Klausel WITH CHECK OPTION darf in folgenden Sichten nicht angegeben werden:

- Sichten, für die die Option READ-ONLY (schreibgeschützte Sicht) definiert wurde.
- Sichten, die auf die Funktion NODENUMBER oder PARTITION, eine nicht deterministische Funktion (z. B. RAND) oder eine Funktion verweisen, für die externe Aktionen ausgeführt werden.
- Typisierte Sichten.

Beispiel 1

Im Folgenden ist ein Beispiel für eine Sichtdefinition aufgeführt, die die Klausel WITH CHECK OPTION verwendet. Diese Option ist erforderlich, um sicherzustellen, dass die Bedingung immer geprüft wird. Die Sicht stellt darüber hinaus sicher, dass für DEPT immer der Wert 10 benutzt wird. Dadurch werden die Eingabewerte für die Spalte DEPT eingeschränkt. Wenn eine Sicht verwendet wird, um einen neuen Wert einzufügen, wird die Verwendung der Klausel WITH CHECK OPTION immer erzwungen:

```
CREATE VIEW EMP_VIEW2
  (EMPNO, EMPNAME, DEPTNO, JOBTITLE, HIREDATE)
AS SELECT ID, NAME, DEPT, JOB, HIREDATE FROM EMPLOYEE
  WHERE DEPT=10
WITH CHECK OPTION;
```

Wird diese Sicht in einer Anweisung INSERT verwendet, wird die Zeile zurückgewiesen, wenn die Spalte DEPTNO nicht den Wert 10 aufweist. Sie sollten unbedingt beachten, dass während der Änderung keine Datenprüfung stattfindet, wenn die Klausel WITH CHECK OPTION nicht angegeben wurde.

Wenn diese Sicht in einer Anweisung SELECT verwendet wird, dann wird die Bedingung (Klausel WHERE) aufgerufen und die dabei generierte Tabelle enthält ausschließlich die Datenzeilen, die mit den angegebenen Werten übereinstimmen. Dies bedeutet, dass die Klausel WITH CHECK OPTION keine Auswirkungen auf das Ergebnis einer Anweisung SELECT hat.

Beispiel 2

Mithilfe einer Sicht können Sie eine Untermenge der Daten einer Tabelle für ein Anwendungsprogramm verfügbar machen und die Daten auf Gültigkeit prüfen, die eingefügt oder aktualisiert werden sollen. Eine Sicht kann Spaltennamen enthalten, die sich von den Namen der entsprechenden Spalten der Originaltabellen unterscheiden. Beispiel:

```
CREATE VIEW <name> (<spalte>, <spalte>, <spalte>)
   SELECT <spaltenname> FROM <tabellenname>
   WITH CHECK OPTION
```

Beispiel 3

Durch die Verwendung von Sichten gestalten sich die Möglichkeiten der Anwendungsprogramme und Endbenutzerabfragen, auf Tabellendaten zuzugreifen, wesentlich flexibler.

Mit der folgenden SQL-Anweisung wird eine Sicht von der Tabelle EMPLOYEE erstellt, die alle Mitarbeiter der Abteilung A00 mit der Personalnummer und der Telefonnummer auflistet:

```
CREATE VIEW EMP_VIEW (DA00NAME, DA00NUM, PHONENO)
   AS SELECT LASTNAME, EMPNO, PHONENO FROM EMPLOYEE
   WHERE WORKDEPT = 'A00'
   WITH CHECK OPTION
```

Die erste Zeile dieser Anweisung benennt die Sicht und definiert ihre Spalten. Der Name EMP_VIEW muss innerhalb seines Schemas in SYSCAT.TABLES eindeutig sein. Der Name der Sicht erscheint als Tabellenname, obwohl die Sicht keine Daten enthält. Die Sicht wird mit den Spalten DA00NAME, DA00NUM und PHONENO definiert, die den Spalten LASTNAME, EMPNO und PHONENO der Tabelle EMPLOYEE entsprechen. Die Spaltennamen werden nacheinander in der aufgeführten Reihenfolge jeweils den Spalten der mit der Anweisung SELECT angegebenen SELECT-Liste zugeordnet. Wenn Spaltennamen nicht angegeben werden, verwendet die Sicht dieselben Namen wie die Spalten der Ergebnistabelle der Anweisung SELECT.

Die zweite Zeile enthält eine Anweisung SELECT, die beschreibt, welche Werte aus der Datenbank ausgewählt werden sollen. Sie kann die Klauseln ALL, DISTINCT, FROM, WHERE, GROUP BY und HAVING enthalten. Der Name bzw. die Namen der Datenobjekte, aus denen die Spalten für die Sicht auszuwählen sind, müssen nach der Klausel FROM angegeben werden.

Beispiel 4

Die Klausel WITH CHECK OPTION gibt an, dass jede in der Sicht aktualisierte oder eingefügte Zeile an der Sichtdefinition überprüft und zurückgewiesen werden muss, wenn sie der Definition nicht entspricht. Dadurch wird die Datenintegrität erhöht, aber auch zusätzlicher Verarbeitungsaufwand verursacht. Wenn diese Klausel nicht angegeben wird, werden Einfügungen und Aktualisierungen nicht an der Sichtdefinition überprüft.

Mit der folgenden SQL-Anweisung wird dieselbe Sicht von der Tabelle EMPLOYEE erstellt, allerdings mit der Klausel SELECT AS:

```
CREATE VIEW EMP_VIEW
   SELECT LASTNAME AS DA00NAME,
          EMPNO AS DA00NUM,
          PHONENO
   FROM EMPLOYEE
   WHERE WORKDEPT = 'A00'
   WITH CHECK OPTION
```

Hier kann die Tabelle EMPLOYEE beispielsweise Gehaltsdaten enthalten, die nicht unbedingt jeder Person zugänglich sein sollten. Die Telefonnummer des Mitarbeiters sollte dagegen allgemein verfügbar sein. In diesem Fall kann eine Sicht erstellt werden, die nur aus den Spalten für den Namen (LASTNAME) und für die Telefonnummer (PHONENO) besteht. Der Zugriff auf die Sicht könnte für die Berechtigung PUBLIC (allgemeiner Zugriff) erteilt werden, während der Zugriff auf die gesamte Tabelle EMPLOYEE auf die Personen beschränkt werden könnte, die die Berechtigung haben, Gehaltsdaten einzusehen.

Definitionen verschachtelter Sichten

Wenn eine Sicht auf einer anderen Sicht basiert, dann richtet sich die Anzahl der Vergleichselemente, die ausgewertet werden müssen, nach der Angabe von WITH CHECK OPTION.

Wenn eine Sicht ohne WITH CHECK OPTION definiert wird, dann wird die Definition der Sicht in keiner Einfüge- oder Aktualisierungsoperation für die Gültig-

keitsprüfung der Daten verwendet. Wenn die Sicht jedoch direkt oder indirekt von einer anderen Sicht abhängt, die mit WITH CHECK OPTION definiert wurde, dann wird die Definition dieser übergeordneten Sicht für die Prüfung aller Einfüge- und Aktualisierungsoperationen verwendet.

Wenn eine Sicht mit WITH CASCADED CHECK OPTION oder nur mit WITH CHECK OPTION (CASCADED ist der Standardwert von WITH CHECK OPTION) definiert wird, dann wird die Definition der Sicht für die Prüfung aller Einfüge- und Aktualisierungsoperationen benutzt. Darüber hinaus übernimmt die Sicht die Suchbedingungen aller aktualisierbaren Sichten, von denen die Sicht abhängig ist. Diese Bedingungen werden auch dann übernommen, wenn diese Sichten die Option WITH CHECK OPTION nicht umfassen. Dann werden die übernommenen Bedingungen multipliziert, um einer Integritätsbedingung zu entsprechen, die für alle Einfüge- oder Aktualisierungsoperationen in Bezug auf die Sicht oder alle anderen Sichten angewendet wird, die von dieser Sicht abhängig sind.

Wenn zum Beispiel die Sicht V2 auf der Sicht V1 basiert und die Prüfoption für V2 mit WITH CASCADED CHECK OPTION definiert ist, werden die Vergleichselemente beider Sichten ausgewertet, wenn für die Sicht V2 INSERT- und UPDATE-Anweisungen ausgeführt werden:

```
CREATE VIEW EMP_VIEW2 AS
  SELECT EMPNO, EMPNAME, DEPTNO FROM EMP
    WHERE DEPTNO = 10
WITH CHECK OPTION;
```

Das folgende Beispiel zeigt eine Anweisung CREATE VIEW, in der WITH CASCADED CHECK OPTION angegeben ist. Die Sicht EMP_VIEW3 wird auf der Basis der Sicht EMP_VIEW2 erstellt, die mit WITH CHECK OPTION erstellt wurde. Wenn Sie einen Datensatz in EMP_VIEW3 einfügen oder dort aktualisieren wollen, dann muss der Datensatz die Werte DEPTNO=10 und EMPNO=20 aufweisen.

```
CREATE VIEW EMP_VIEW3 AS
  SELECT EMPNO, EMPNAME, DEPTNO FROM EMP_VIEW2
    WHERE EMPNO > 20
WITH CASCADED CHECK OPTION;
```

Anmerkung: Die Verwendung der Bedingung DEPTNO=10 wird für Einfüge- oder Aktualisierungsoperationen in EMP_VIEW3 auch dann durchgesetzt, wenn EMP_VIEW2 die Angabe WITH CHECK OPTION nicht enthält.

Die Option WITH LOCAL CHECK OPTION kann auch bei der Erstellung einer Sicht angegeben werden. Wenn für eine Sicht LOCAL CHECK OPTION definiert wurde, dann wird die Definition der Sicht für die Prüfung aller Einfüge- und Aktualisierungsoperationen verwendet. Die Sicht übernimmt allerdings nicht die Suchbedingungen der aktualisierbaren Sichten, von denen diese abhängig ist.

Löschfähige Sichten

Abhängig von der Art und Weise, in der eine Sicht definiert ist, kann die zum Löschen verwendet werden. Für eine solche löschfähige Sicht kann die Anweisung DELETE erfolgreich ausgeführt werden.

Eine löschfähige Sicht muss die folgenden Regeln erfüllen:

- Jede Klausel FROM des Outer Fullselect identifiziert nur eine Tabelle (ohne die Klausel OUTER), eine löschfähige Sicht (ohne die Klausel OUTER), einen löschfähigen verschachtelten Tabellenausdruck oder löschfähigen allgemeinen Tabellenausdruck.

- Der Datenbankmanager muss in der Lage sein, die in der Tabelle zu löschenden Zeilen mithilfe der Sichtdefinition abzuleiten. Bestimmte Operationen verhindern dies jedoch.
  - Eine Zusammenfassung mehrerer Zeilen zu einer Gruppe mithilfe der Klausel GROUP BY oder bestimmter Spaltenfunktionen führt zum Verlust der ursprünglichen Zeile und dazu, dass die Sicht nicht zum Löschen verwendet werden kann.
  - Ebenso ist keine Tabelle vorhanden, aus der gelöscht werden kann, wenn die Zeilen aus einer Klausel VALUES abgeleitet werden. Auch in diesem Fall ist die Sicht nicht löschfähig.
- Der Outer Fullselect verwendet die Klauseln GROUP BY und HAVING nicht.
- Die SELECT-Liste des Outer Fullselect umfasst keine Spaltenfunktionen.
- Der Outer Fullselect verwendet keine Gruppenoperationen (UNION, EXCEPT oder INTERSECT). Eine Ausnahme bildet hierbei UNION ALL.
- Die Tabellen in den Operanden von UNION ALL dürfen nicht gleich sein, und jeder Operand muss löschbar sein.
- Die SELECT-Liste des Outer Fullselect enthält keine Klausel DISTINCT.

Eine Sicht muss alle oben aufgeführten Regeln erfüllen, um als löschfähige Sicht eingestuft zu werden. Die folgende Sicht ist z. B. löschfähig. Sie erfüllt alle Regeln für löschfähige Sichten.

```
CREATE VIEW löschfähige_sicht
  (number, date, start, end)
AS
  SELECT number, date, start, end
  FROM employee.summary
  WHERE date='01012007'
```

Einfügefähige Sichten

Einfügefähige Sichten ermöglichen Ihnen das Einfügen von Zeilen mithilfe der Sichtdefinition. Eine Sicht wird als einfügefähig bezeichnet, wenn ein INSTEAD OF-Trigger für die Einfügeoperation für die Sicht definiert wurde oder wenn mindestens eine Spalte der Sicht (unabhängig von einem INSTEAD OF-Trigger für die Aktualisierung) aktualisierbar ist und die Fullselect-Anweisung der Sicht keine Angabe UNION ALL enthält. Eine bestimmte Zeile kann in eine Sicht (einschließlich einer UNION ALL-Sicht) nur dann eingefügt werden, wenn sie die Prüfbedingungen von genau einer der zugrunde liegenden Tabellen erfüllt. Um eine Einfügung in eine Sicht vorzunehmen, die nicht aktualisierbare Spalten enthält, dürfen diese Spalten in der Spaltenliste nicht aufgeführt werden.

Das folgende Beispiel zeigt eine einfügefähige Sicht. In diesem Beispiel schlägt die Einfügung der Sicht jedoch fehl. Dies ist darauf zurückzuführen, dass Spalten in der Tabelle enthalten sind, die keine Nullwerte akzeptieren. Einige dieser Spalten sind in der Sichtdefinition nicht vorhanden. Wenn Sie versuchen, mit der Sicht einen Wert einzufügen, versucht der Datenbankmanager, einen Nullwert in eine Spalte mit dem Merkmal NOT NULL einzufügen. Diese Aktion ist nicht zulässig.

```
CREATE VIEW einfügefähige_sicht
  (number, name, quantity)
AS
SELECT number, name, quantify FROM ace.supplies
```

Anmerkung: Die für die Tabelle definierten Integritätsbedingungen gelten unabhängig von den Operationen, die mit einer auf dieser Tabelle basierenden Sicht ausgeführt werden können.

Aktualisierungsfähige Sichten

Eine aktualisierungsfähige Sicht ist ein Sonderfall einer löschfähigen Sicht. Eine löschfähige Sicht wird zu einer aktualisierungsfähigen Sicht, wenn zumindest eine ihrer Spalten aktualisiert werden kann.

Eine Spalte einer Sicht ist aktualisierbar, wenn alle folgenden Regeln erfüllt werden:

- Die Sicht ist löschfähig.
- Die Spalte kann (ohne eine Dereferenzierungsoperation) in eine Spalte einer Tabelle aufgelöst werden und die Option READ ONLY wurde nicht angegeben.
- Alle zugehörigen Spalten der Operanden von UNION ALL verfügen über exakt übereinstimmende Datentypen (einschließlich Länge oder Genauigkeit und Maßstab) und übereinstimmende Standardwerte, wenn der Fullselect der Sicht UNION ALL umfasst.

Im folgenden Beispiel werden konstante Werte verwendet, die nicht aktualisiert werden können. Die Sicht ist jedoch löschfähig und mindestens eine der Spalten kann aktualisiert werden. Aus diesem Grund handelt es sich um eine aktualisierungsfähige Sicht.

```
CREATE VIEW aktualisierungsfähige_sicht
  (number, current_date, current_time, temperature)
AS
  SELECT number, CURRENT DATE, CURRENT TIME, temperature)
FROM weather.forecast
WHERE number = 300
```

Schreibgeschützte Sichten

Eine Sicht wird als *schreibgeschützt* bezeichnet, wenn sie *nicht* löschfähig, aktualisierungsfähig oder einfügefähig ist. Eine Sicht kann auch schreibgeschützt sein, wenn sie nicht mindestens einer der Regeln für löschfähige Sichten entspricht.

Die Spalte READONLY in der Katalogsicht SYSCAT.VIEWS gibt eine schreibgeschützte Sicht (R = Read-only) an.

Das folgende Beispiel zeigt keine löschfähige Sicht, weil die Klausel DISTINCT verwendet wird und die SQL-Anweisung mehr als eine Tabelle betrifft:

```
CREATE VIEW schreibgeschützte_sicht
  (name, phone, address)
AS
SELECT DISTINCT viewname, viewphone, viewaddress
FROM employee.history adam, employer.dept sales
WHERE adam.id = sales.id
```

Erstellen von Sichten

Sichten werden aus einer oder mehreren Tabellen, Kurznamen oder Sichten abgeleitet und können beim Abrufen von Daten frei austauschbar mit Tabellen verwendet werden. Wenn Änderungen an den in einer Sicht gezeigten Daten vorgenommen werden, werden die Daten in der Tabelle selbst geändert. Die Tabelle, der Kurzname bzw. die Sicht, auf der die Sicht basieren soll, muss vorhanden sein, bevor die Sicht erstellt werden kann.

Informationen zu diesem Vorgang

Eine Sicht kann zur Einschränkung des Zugriffs auf sensible Daten verwendet werden, während auf andere Daten ein allgemeinerer Zugriff zugelassen wird.

Beim Einfügen in eine Sicht, in der eine SELECT-Liste der Sichtdefinition direkt oder indirekt den Namen einer Identitätsspalte einer Tabelle umfasst, gelten dieselben Regeln, wie wenn die INSERT-Anweisung direkt auf die Identitätsspalte der Tabelle verweisen würde.

Über die bereits beschriebene Verwendung hinaus kann eine Sicht auch folgenden Zwecken dienen:

- Ändern einer Tabelle ohne Auswirkungen auf Anwendungsprogramme. Dies kann durch das Erstellen einer Sicht auf der Basis einer zugrunde liegenden Tabelle geschehen. Anwendungen, die die zugrunde liegende Tabelle verwenden, werden durch die Erstellung der neuen Sicht nicht beeinflusst. Neue Anwendungen können die erstellte Sicht für andere Zwecke verwenden als jene Anwendungen, die die zugrunde liegende Tabelle verwenden.
- Summieren der Werte in einer Spalte, Auswählen der größten Werte oder Berechnen der Durchschnittswerte.
- Bereitstellen des Zugriffs auf Informationen an einer oder mehreren Datenquellen. Sie können innerhalb der Anweisung CREATE VIEW auf Kurznamen verweisen sowie Sichten für mehrere Positionen bzw. globale Sichten (die Sicht könnte Informationen aus mehreren Datenquellen auf verschiedenen Systemen verknüpfen) erstellen.

  Wenn Sie eine Sicht erstellen, die mit der Standardsyntax von CREATE VIEW auf Kurznamen verweist, erscheint eine Warnung, die darauf hinweist, dass die Berechtigungs-ID der Benutzer dieser Sicht für den Zugriff auf die zugrunde liegenden Objekte an den Datenquellen verwendet werden und nicht die Berechtigungs-ID des Erstellers dieser Sicht. Diese Warnung können Sie mit dem Schlüsselwort FEDERATED unterdrücken.

Eine typisierte Sicht basiert auf einem vordefinierten strukturierten Typ. Sie können mithilfe der Anweisung CREATE VIEW eine typisierte Sicht erstellen.

Eine Alternative zur Erstellung einer Sicht ist die Verwendung eines verschachtelten oder allgemeinen Tabellenausdrucks, um das Durchsuchen von Katalogen zu reduzieren und die Leistung zu erhöhen.

Das folgende Beispiel zeigt eine Anweisung CREATE VIEW. Die zugrunde liegende Tabelle EMPLOYEE enthält Spalten mit den Namen SALARY und COMM. Aus Sicherheitsgründen wird diese Sicht aus den Spalten ID, NAME, DEPT, JOB und HIREDATE erstellt. Darüber hinaus wird der Zugriff auf die Spalte DEPT eingeschränkt. Diese Definition zeigt nur die Informationen zu Mitarbeitern an, die zu der Abteilung mit der Abteilungsnummer (DEPTNO) 10 gehören.

```
CREATE VIEW EMP_VIEW1
  (EMPID, EMPNAME, DEPTNO, JOBTITLE, HIREDATE)
  AS SELECT ID, NAME, DEPT, JOB, HIREDATE FROM EMPLOYEE
    WHERE DEPT=10;
```

Nachdem die Sicht definiert wurde, können die Zugriffsberechtigungen angegeben werden. Dadurch wird die Datensicherheit gewährleistet, da nur Zugriff auf eine eingeschränkte Sicht der Tabelle besteht. Wie im obigen Beispiel dargestellt, kann

eine Sicht eine Klausel WHERE enthalten, um den Zugriff auf bestimmte Zeilen einzuschränken, oder eine Untergruppe der Spalten, um den Zugriff auf bestimmte Datenspalten einzuschränken.

Die Spaltennamen in der Sicht müssen nicht mit den Spaltennamen der Tabelle übereinstimmen. Der Tabellenname verfügt wie der Sichtname über ein zugeordnetes Schema.

Nachdem die Sicht definiert wurde, kann diese in Anweisungen wie zum Beispiel SELECT, INSERT, UPDATE und DELETE (mit Einschränkungen) verwendet werden. Der Datenbankadministrator kann entscheiden, ob einer bestimmten Benutzergruppe umfangreichere Zugriffsrechte für die Sicht gewährt werden sollen als für die Tabelle.

Erstellen von Sichten mit benutzerdefinierten Funktionen (UDFs)

Wenn Sie eine Sicht erstellen, die mit benutzerdefinierten Funktionen (UDFs = User-Defined Functions) arbeitet, dann verwendet diese Sicht während ihrer gesamten Lebensdauer die für sie definierte UDF. Dies gilt auch dann, wenn Sie zu einem späteren Zeitpunkt andere UDFs mit dem gleichen Namen erstellen. Wenn Sie eine neue UDF verwenden möchten, müssen Sie die Sicht erneut erstellen.

Informationen zu diesem Vorgang

Mit der folgenden SQL-Anweisung wird eine Sicht mit einer Funktion in ihrer Definition erstellt:

```
CREATE VIEW EMPLOYEE_PENSION (NAME, PENSION)
  AS SELECT NAME, PENSION(HIREDATE,BIRTHDATE,SALARY,BONUS)
  FROM EMPLOYEE
```

Die benutzerdefinierte Funktion PENSION berechnet die aktuellen Pensionsansprüche eines Mitarbeiters mit einer Formel, die mit den Werten des Einstellungsdatums (HIREDATE), des Geburtsdatums (BIRTHDATE), des Gehalts (SALARY) und der Sondervergütungen (BONUS) arbeitet.

Ändern typisierter Sichten

Bestimmte Eigenschaften einer typisierten Sicht können geändert werden, ohne dass hierzu die Sicht gelöscht und anschließend erneut erstellt werden muss. Eine dieser Eigenschaften bezieht sich auf das Hinzufügen eines Bereichs zu einer Verweisspalte einer typisierten Sicht.

Informationen zu diesem Vorgang

Die Anweisung ALTER VIEW ändert eine vorhandene Definition einer typisierten Sicht durch Ändern einer Verweistypspalte zum Hinzufügen eines Bereichs. Die Anweisung DROP löscht eine typisierte Sicht. Außerdem können Sie folgende Operationen ausführen:

- Ändern des Inhalts einer typisierten Sicht mithilfe von INSTEAD OF-Triggern
- Ändern einer typisierten Sicht zum Aktivieren der Statistikerfassung

Für Änderungen, die Sie an dem Inhalt vornehmen, der einer typisierten Sicht zugrunde liegt, benötigen Sie Trigger. Andere Änderungen an einer typisierten Sicht können Sie nur durchführen, indem Sie die typisierte Sicht löschen und erneut erstellen.

Der Datentyp des Spaltennamens in der Anweisung ALTER VIEW muss REF sein (Typ des Namens der typisierten Tabelle oder des Namens der typisierten Sicht).

Vorgehensweise

Setzen Sie die Anweisung ALTER VIEW ab, um eine typisierte Sicht über die Befehlszeile zu ändern. Beispiel:

```
ALTER VIEW sichtname ALTER spaltenname
   ADD SCOPE name_der_typisierten_tabelle_oder_sicht
```

Ergebnisse

Andere Datenbankobjekte wie Tabellen und Indizes sind nicht betroffen, obwohl Pakete und im Cache zwischengespeicherte dynamische Anweisungen als ungültig markiert werden.

Recovery funktionsunfähiger Sichten

Als funktionsunfähige Sicht wird eine Sicht bezeichnet, die für SQL-Anweisungen nicht mehr zur Verfügung steht.

Informationen zu diesem Vorgang

Sichten können aufgrund folgender Ursachen *funktionsunfähig* werden:

- Infolge eines widerrufenen Zugriffsrechts für eine zugrunde liegende Tabelle.
- Wenn eine Tabelle, ein Aliasname oder eine Funktion gelöscht wurde.
- Wenn die übergeordnete Sicht funktionsunfähig wird. (Eine übergeordnete Sicht ist eine typisierte Sicht, auf der eine andere typisierte Sicht, d. h. eine untergeordnete Sicht, basiert.)
- Wenn die Sichten, von denen die Sichten abhängen, gelöscht werden.

Wenn Sie eine funktionsunfähige Sicht nicht wiederherstellen möchten, können Sie sie explizit mit der Anweisung DROP VIEW löschen, oder Sie können eine Sicht mit demselben Namen, aber einer anderen Definition erstellen.

Eine funktionsunfähige Sicht hat nur noch Einträge in den Katalogsichten SYSCAT.TABLES und SYSCAT.VIEWS. Alle Einträge in den Katalogsichten SYSCAT.TABDEP, SYSCAT.TABAUTH, SYSCAT.COLUMNS und SYSCAT.COLAUTH werden entfernt.

Vorgehensweise

Eine Recovery für eine funktionsunfähige Sicht kann auf folgende Weise durchgeführt werden:

1. Stellen Sie fest, mit welcher SQL-Anweisung die Sicht zu Anfang erstellt wurde. Diese Information können Sie der Spalte TEXT der Katalogsicht SYSCAT.VIEW entnehmen.
2. Definieren Sie das aktuelle Schema für den Inhalt der Spalte QUALIFIER.
3. Definieren Sie den Funktionspfad für den Inhalt der Spalte FUNC_PATH.
4. Erstellen Sie die Sicht erneut, indem Sie die Anweisung CREATE VIEW mit demselben Sichtnamen und derselben Definition verwenden.
5. Verwenden Sie die Anweisung GRANT, um alle Zugriffsrechte, die zuvor für die Sicht erteilt waren, erneut zu erteilen. (Beachten Sie, dass alle für eine funktionsunfähig gewordene Sicht erteilten Zugriffsrechte widerrufen werden.)

Löschen von Sichten

Verwenden Sie zum Löschen von Sichten die Anweisung DROP VIEW. Alle Sichten, die von der gelöschten Sicht abhängig sind, werden funktionsunfähig.

Vorgehensweise

Geben Sie in die Befehlszeile Folgendes ein, um eine Sicht zu löschen:

```
DROP VIEW sichtname
```

Beispiel

Das folgende Beispiel zeigt, wie die Sicht EMP_VIEW gelöscht wird:

```
DROP VIEW EMP_VIEW
```

Ebenso wie bei Tabellenhierarchien kann eine gesamte Sichtenhierarchie in einer einzigen Anweisung gelöscht werden, in der die Stammsicht der Hierarchie angegeben wird, wie im folgenden Beispiel gezeigt:

```
DROP VIEW HIERARCHY VPerson
```

Kapitel 18. Nutzungslisten

Eine *Nutzungsliste* ist ein Datenbankobjekt, das die DML-Anweisungsabschnitte aufzeichnet, die auf eine bestimmte Tabelle oder einen bestimmten Index verweisen, und während der Ausführung eines Abschnitts Statistikdaten dazu erfasst, welche Auswirkungen der Abschnitt auf die einzelnen Objekte hat. Mithilfe von Nutzungslisten können Sie ermitteln, welche DML-Anweisungen sich auf eine Tabelle oder einen Index ausgewirkt haben.

In einer Nutzungsliste werden nur Daten erfasst, wenn die Nutzungsliste aktiv ist. Die einzelnen Einträge in einer Nutzungsliste enthalten Daten für jeden DML-Anweisungsabschnitt, der auf die Tabelle oder den Index verweist, für die bzw. den die Nutzungsliste erstellt wurde. Jeder Eintrag enthält Informationen dazu, wie häufig der Abschnitt ausgeführt wurde, sowie kumulative Statistikdaten, die angeben, wie sich der Abschnitt im Verlauf aller Ausführungen auf die Tabelle bzw. den Index ausgewirkt hat.

Verweise in der Liste können anhand der Werte kumuliert werden, die in der folgenden Tabelle aufgelistet sind.

Tabelle 79. Kumulierte Werte für Nutzungslistenverweise

Wert	Beschreibung
executable_ID	Identifiziert die ausgeführte SQL-Anweisung.
mon_interval_ID	Identifiziert das Überwachungsintervall an dem Zeitpunkt, an dem der Wert *executable_ID* zur Nutzungsliste hinzugefügt wurde.

Betrachten Sie das folgende Beispiel für die Verwendung von Nutzungslisten. Bei einem routinemäßigen Überwachungsvorgang sehen Sie einen hohen Wert für das Monitorelement **rows_read** für eine bestimmte Tabelle in der Ausgabe für die Tabellenfunktion MON_GET_TABLE. Sie können eine Nutzungsliste für die Tabelle verwenden, um zu ermitteln, welche DML-Anweisungen zu dem hohen Wert beigetragen haben. Wenn Sie feststellen, dass ein Problem besteht, können Sie anhand der Statistikdaten in der Nutzungsliste ermitteln, welche Anweisungen insbesondere weitere Überwachung oder Optimierung erfordern.

Sie können mehrere Nutzungslisten für eine Tabelle oder einen Index erstellen. Das gleichzeitige Aktivieren von mehreren Nutzungslisten kann sich jedoch nachteilig auf die Datenbankleistung und die Speicherbelegung auswirken.

Einschränkungen

Die folgenden Einschränkungen gelten für Nutzungslisten:

- Nutzungslisten können nur Informationen zu DML-Anweisungen erfassen.
- Nutzungslisten können nur für nicht typisierte Tabellen erstellt werden. Die folgenden Tabellentypen und Objekte werden nicht unterstützt:
  - Aliasnamen
  - Erstellte temporäre Tabellen
  - Tabellen mit aufgehobener Zuordnung

  - Hierarchietabellen
  - Kurznamen
  - Typisierte Tabellen
  - Sichten
- Nutzungslisten können nur für die folgenden Indextypen erstellt werden:
  - Blockindizes
  - Clusterindizes
  - Dimensionsblockindizes
  - Reguläre Indizes
- Das Dienstprogramm **db2look** extrahiert nicht die DDL-Anweisungen, die zum Erstellen von Kopien von Nutzungslisten erforderlich sind.

Hinweise zum Speicher für Nutzungslisten und Prüfungsabhängigkeiten

Nach der Aktivierung einer Nutzungsliste ordnet der Datenbankmanager Speicher zu, damit die erfassten Daten gespeichert werden, wenn ein Abschnitt zum ersten Mal auf das Objekt verweist, für das die Nutzungsliste definiert wurde. Während der Lebensdauer der Nutzungsliste können verschiedene Aktionen Auswirkungen auf diesen Speicher haben und/oder die Nutzungsliste inaktivieren.

Es gibt folgende allgemeine Hinweise zum Hauptspeicher:

- ***Hinweise zur Größe der Nutzungsliste:*** Wählen Sie eine angemessene Listengröße aus oder legen Sie für den Konfigurationsparameter **mon_heap_sz** den Wert AUTOMATIC fest, sodass die Größe des Zwischenspeichers für die Überwachung vom Datenbankmanager verwaltet wird.
- ***Leistungsaspekte:*** Erstellen Sie zum Beibehalten einer hohen Leistung Nutzungslisten, deren Größe auf den Umfang eingeschränkt ist, der zum Erfassen der für Sie erforderlichen Informationen benötigt wird. Für jede Nutzungsliste ist Systemspeicher erforderlich. Die Systemleistung kann sich aufgrund der Aktivierung zusätzlicher Nutzungslisten vermindern.

Die folgende Tabelle zeigt im Einzelnen, wie sich verschiedene Aktionen auf den zugeordneten Speicher auswirken.

Aktion	Auswirkungen	Auswirkungen, wenn sich die Nutzungsliste auf eine partitionierte Tabelle oder einen Index bezieht	Auswirkungen in einer Umgebung mit partitionierten Datenbanken- oder DB2 pureScale-Umgebung
Nach der erstmaligen Aktivierung einer Nutzungsliste verweist ein Abschnitt auf das Objekt, für das die Nutzungsliste definiert wurde.	Für die Nutzungsliste wird Speicher zugeordnet.	Für jede Datenpartition wird Speicher zugeordnet. Beispiel: Wenn die Nutzungsliste eine Speichermenge von 2 MB erfordert und drei Datenpartitionen vorhanden sind, wird eine Gesamtspeichermenge von 6 MB zugeordnet.	Der Speicher wird für jede Teilkomponente zugeordnet. Beispiel: Wenn die Nutzungsliste eine Speichermenge von 2 MB erfordert und drei Teilkomponenten vorhanden sind, wird eine Gesamtspeichermenge von 6 MB zugeordnet.
Die Größe der Nutzungsliste kann geändert werden.	Die der Nutzungsliste zugeordnete Speichermenge ändert sich bei der nächsten Aktivierung der Nutzungsliste.	Die der Nutzungsliste für die einzelnen Datenpartitionen zugeordnete Speichermenge ändert sich bei der nächsten Aktivierung der Nutzungsliste.	Die der Nutzungsliste für die einzelnen Teilkomponenten zugeordnete Speichermenge ändert sich bei der nächsten Aktivierung der Nutzungsliste.

Aktion	Auswirkungen	Auswirkungen, wenn sich die Nutzungsliste auf eine partitionierte Tabelle oder einen Index bezieht	Auswirkungen in einer Umgebung mit partitionierten Datenbanken- oder DB2 pureScale-Umgebung
Der Tabelle oder dem Index, für die/den die Nutzungsliste definiert ist, wird eine neue Datenpartition hinzugefügt oder zugeordnet.	Nicht zutreffend	Für die neue Datenpartition wird Speicher zugeordnet, wenn ein Abschnitt das nächste Mal auf die Tabelle oder den Index verweist.	Nicht zutreffend
Die Nutzungsliste wird gelöscht.	Der für die Nutzungsliste zugeordnete Speicher wird freigegeben.	Der für die Nutzungsliste zugeordnete Speicher wird für alle Datenpartitionen freigegeben.	Der für die Nutzungsliste zugeordnete Speicher wird für alle Teilkomponenten freigegeben.
Die Tabelle oder der Index, für die/den die Nutzungsliste definiert wurde, wird gelöscht.	Der für die Nutzungsliste zugeordnete Speicher wird freigegeben und der Katalogeintrag für die Nutzungsliste wird inaktiviert. Der Katalogeintrag kann mithilfe der Prozedur ADMIN_REVALIDATE_ DB_OBJECTS erneut aktiviert werden.	Der für die Nutzungsliste zugeordnete Speicher wird für alle Datenpartitionen freigegeben und der Katalogeintrag für die Nutzungsliste wird inaktiviert. Der Katalogeintrag kann mithilfe der Prozedur ADMIN_REVALIDATE_ DB_OBJECTS erneut aktiviert werden.	Der für die Nutzungsliste zugeordnete Speicher wird für alle Teilkomponenten freigegeben und der Katalogeintrag für die Nutzungsliste wird inaktiviert. Der Katalogeintrag kann mithilfe der Prozedur ADMIN_REVALIDATE_ DB_OBJECTS erneut aktiviert werden.
Die Instanz oder Datenbank wird inaktiviert.	Der für die Nutzungsliste zugeordnete Speicher wird freigegeben.	Der für die Nutzungsliste zugeordnete Speicher wird für alle Datenpartitionen freigegeben.	Der für die Nutzungsliste zugeordnete Speicher wird für alle Teilkomponenten freigegeben.
Mit der Anweisung SET USAGE LIST STATE wird der Speicher freigegeben, der für die Nutzungsliste zugeordnet ist.	Der für die Nutzungsliste zugeordnete Speicher wird freigegeben.	Der für die Nutzungsliste zugeordnete Speicher wird für alle Datenpartitionen freigegeben.	Der für die Nutzungsliste zugeordnete Speicher wird für alle Teilkomponenten freigegeben.
Die Zuordnung einer Datenpartition zu der Tabelle oder dem Index, für die/den die Nutzungsliste erstellt wurde, wird aufgehoben.	Nicht zutreffend	Der Speicher für die Datenpartition, für die die Zuordnung aufgehoben wurde, wird freigegeben.	Nicht zutreffend
Eine Datenbankteilkomponente wird gelöscht oder inaktiviert.	Nicht zutreffend	Nicht zutreffend	Der für die gelöschte oder inaktivierte Teilkomponente zugeordnete Speicher wird freigegeben.

Teil 4. Referenz

Kapitel 19. Namenskonventionen

Allgemeine Namenskonventionen

Für die Benennung aller Datenbankobjekte, Benutzernamen, Kennwörter, Gruppen, Dateien und Pfade gelten Regeln. Einige dieser Regeln sind für die Plattform spezifisch, auf der Sie arbeiten.

Die folgenden Punkte sind zum Beispiel in Bezug auf die Verwendung von Groß- und Kleinbuchstaben in den Namen von Objekten zu beachten, die im Dateisystem (Datenbanken, Instanzen usw.) sichtbar sind:

- Auf UNIX-Plattformen wird bei Namen die Groß-/Kleinschreibung unterschieden. Zum Beispiel ist ein Verzeichnis `/data1` nicht mit den Verzeichnissen `/DATA1` oder `/Data1` identisch.
- Auf Windows-Plattformen wird die Groß-/Kleinschreibung in Namen nicht unterschieden. Zum Beispiel bezeichnet `\data1` denselben Namen wie `\DATA1` und `\Data1`.

Wenn nichts anderes angegeben ist, dürfen alle Namen die folgenden Zeichen enthalten:

- Die Buchstaben A - Z und a bis z, wie sie im ASCII-Grundzeichensatz (7-Bit-Zeichensatz) definiert sind. Wenn sie in Kennungen für Objekte verwendet werden, die durch SQL-Anweisungen erstellt werden, werden die Kleinbuchstaben “a” bis “z” in die entsprechenden Großbuchstaben umgewandelt, wenn sie nicht in Anführungszeichen (“) gesetzt werden.
- 0 bis 9.
- ! % ( ) { } . - ^ ~ _ (Unterstreichungszeichen), @, #, $ und Leerzeichen.
- \ (umgekehrter Schrägstrich oder Backslash).

Einschränkungen

- Namen dürfen nicht mit einer Ziffer oder dem Unterstreichungszeichen beginnen.
- Für SQL reservierte Wörter dürfen nicht als Namen von Tabellen, Sichten, Spalten, Indizes oder Berechtigungs-IDs verwendet werden.
- In Namen für Verzeichnis- oder Dateinamen sollten nur die Buchstaben verwendet werden, die im ASCII-Grundzeichensatz definiert sind. Das Betriebssystem Ihres Computers unterstützt vielleicht verschiedene Codepages, jedoch ist es möglich, dass andere Zeichen als ASCII-Zeichen nicht zuverlässig funktionieren. Die Verwendung von Nicht-ASCII-Zeichen kann insbesondere in einer verteilten Umgebung, in der verschiedene Computer unterschiedliche Codepages verwenden, zu einem Problem werden.
- In Abhängigkeit vom verwendeten Betriebssystem und dem Ort, an dem Sie mit der DB2-Datenbank arbeiten, sind möglicherweise weitere Sonderzeichen verfügbar, die ebenfalls verwendet werden können. Diese Zeichen können funktionieren, jedoch gibt es keine Garantie, dass sie funktionieren. Daher wird empfohlen, beim Benennen von Objekten in der Datenbank keine anderen als die oben aufgelisteten Sonderzeichen zu verwenden.

- Benutzer- und Gruppennamen müssen auch die Regeln einhalten, die durch bestimmte Betriebssysteme festgelegt werden. Zum Beispiel sind auf Linux- und UNIX-Plattformen nur die folgenden Zeichen für Benutzernamen und Gruppennamen zulässig: a bis z in Kleinbuchstaben, 0 bis 9 und _ (Unterstreichungszeichen) für Namen, die nicht mit einer Ziffer zwischen 0 und 9 beginnen.
- Längen dürfen die in „SQL- und XML-Begrenzungen" (SQL and XML limits) im Handbuch *SQL Reference* aufgeführten Längen nicht überschreiten.
- **Einschränkungen zur Berechtigungs-ID AUTHID:** In DB2 Version 9.5 und höher kann die Berechtigungs-ID eine Größe von 128 Byte haben. Wenn die Berechtigungs-ID jedoch als Benutzer-ID oder Gruppenname des Betriebssystems interpretiert wird, gelten die Einschränkungen des Betriebssystems. Benutzer-IDs auf Linux- und UNIX-Betriebssystemen dürfen bis zu 8 Zeichen enthalten. Benutzer-IDs auf Windows-Betriebssystemen dürfen bis zu 30 Zeichen für Benutzer-IDs und Gruppennamen enthalten. Daher können Sie zwar eine Berechtigungs-ID mit einer Größe von 128-Byte erteilen, jedoch ist es nicht möglich, als Benutzer, der diese Berechtigungs-ID hat, eine Verbindung herzustellen. Wenn Sie ein eigenes Sicherheits-Plug-in schreiben, können Sie die erweiterten Größen für die Berechtigungs-ID verwenden. Zum Beispiel können Sie Ihr Sicherheits-Plug-in mit einer Benutzer-ID mit einer Größe von 30-Byte versehen, und das Plug-in gibt während der Authentifizierung eine 128-Byte-Berechtigungs-ID zurück, mit der Sie eine Verbindung herstellen können.

Darüber hinaus müssen Sie auch die Namenskonventionen der einzelnen Objekte, die Namenskonventionen in einer für einen kulturübergreifenden Einsatz geeigneten Umgebung und die Namenskonventionen in einer Unicode-Umgebung beachten.

Namenskonventionen für DB2-Objekte

Für alle Objekte müssen die allgemeinen Namenskonventionen beachtet werden. Darüber hinaus gelten für bestimmte Objekte zusätzliche Einschränkungen, die den folgenden Tabellen zu entnehmen sind.

Tabelle 80. Namenskonventionen für Datenbanken, Aliasnamen von Datenbanken und Instanzen

Objekte	Richtlinien
• Datenbanken • Aliasnamen von Datenbanken • Instanzen	• Datenbanknamen müssen an der Position, an der sie katalogisiert werden, eindeutig sein. Bei Linux- und UNIX-Implementierungen ist diese Position ein Verzeichnispfad. Bei Windows-Implementierungen ist diese Position eine logische Platte. • Aliasnamen von Datenbanken müssen innerhalb des Systemdatenbankverzeichnisses eindeutig sein. Beim Erstellen einer neuen Datenbank wird der Aliasname der Datenbank standardmäßig so definiert, dass er mit dem Datenbanknamen identisch ist. Daher können Sie keine Datenbank mit einem Namen erstellen, der bereits als Aliasname einer Datenbank verwendet wird, auch wenn noch keine Datenbank mit diesem Namen vorhanden ist. • Datenbanknamen, Datenbankaliasnamen und Instanznamen dürfen die Länge von 8 Byte nicht überschreiten. • Unter Windows dürfen Instanzen keine Namen haben, die als Namen von Services (Diensten) verwendet werden. **Anmerkung:** Um mögliche Probleme zu vermeiden, empfiehlt es sich, die Sonderzeichen @, # und $ nicht in Datenbanknamen zu verwenden, wenn die Datenbank in einer Übertragungsumgebung verwendet werden soll. Darüber hinaus sollten Sie diese Zeichen sowie Umlaute nicht benutzen, wenn Sie die Datenbank in einer anderen Sprache verwenden wollen, da diese Zeichen nicht auf allen Tastaturen in gleicher Weise verfügbar sind.

Tabelle 81. Namenskonventionen für Datenbankobjekte

Objekte	Richtlinien
• Aliasnamen • Prüfrichtlinien • Pufferpools • Spalten • Ereignismonitore • Indizes • Methoden • Knotengruppen • Pakete • Paketversionen • Rollen • Schemata • Gespeicherte Prozeduren • Tabellen • Tabellenbereiche • Trigger • Gesicherter Kontext • Benutzerdefinierte Funktionen • Benutzerdefinierte Typen • Sichten	• Die Längen für Kennungen dieser Objekte dürfen die in „SQL- und XML-Begrenzungen" (SQL and XML limits) im Handbuch *SQL Reference* aufgeführten Längen nicht überschreiten. Objektnamen dürfen auch die folgenden Elemente enthalten: – Gültige Zeichen mit Akzent und Umlaute (wie beispielsweise ö) – Mehrbytezeichen mit Ausnahme von Mehrbyteleerzeichen (in Mehrbyteumgebungen) • Paketnamen und Paketversionen dürfen auch Punkte (.), Bindestriche (-) und Doppelpunkte (:) enthalten. Weitere Informationen finden Sie im Abschnitt zu „Kennungen" (Identifiers) im Handbuch *SQL Reference*.

Tabelle 82. Namenskonventionen für Objekte in föderierten Datenbanken

Objekte	Richtlinien
• Funktionszuordnungen • Indexspezifikationen • Kurznamen • Server • Typenzuordnungen • Benutzerzuordnungen • Wrapper	Die Längen dieser Objekte dürfen die in „SQL- und XML-Begrenzungen" (SQL and XML limits) im Handbuch *SQL Reference* aufgeführten Längen nicht überschreiten. Namen für Objekte in föderierten Datenbanken dürfen auch die folgenden Elemente enthalten: • Gültige Zeichen mit Akzent und Umlaute (wie beispielsweise ö) • Mehrbytezeichen mit Ausnahme von Mehrbyteleerzeichen (in Mehrbyteumgebungen)

Namen von begrenzten Bezeichnern und Objekten

Schlüsselwörter dürfen verwendet werden. Wird ein Schlüsselwort in einem Kontext verwendet, in dem es auch als SQL-Schlüsselwort interpretiert werden kann, muss es als begrenzter Bezeichner angegeben werden.

Mithilfe der begrenzten Bezeichner ist es möglich, ein Objekt zu erstellen, dessen Name gegen diese Namenskonventionen verstößt. Bei der späteren Verwendung eines solchen Objekts können jedoch Fehler auftreten. Wenn Sie zum Beispiel eine Spalte mit einem Namen erstellt haben, in dem ein Pluszeichen (+) oder ein Minuszeichen (-) vorkommt, und Sie diese Spalte später in einem Index verwenden, treten Probleme auf, wenn Sie versuchen, die Tabelle zu reorganisieren.

Weitere Informationen zu Schemanamen

- Benutzerdefinierte Typen (UDTs) dürfen keine Schemanamen verwenden, deren Längen die in „SQL- und XML-Begrenzungen" (SQL and XML limits) im Handbuch *SQL Reference* aufgeführten Längen überschreiten.
- Die folgenden Schemanamen sind reservierte Wörter und dürfen nicht verwendet werden: SYSCAT, SYSFUN, SYSIBM, SYSSTAT, SYSPUBLIC.
- Um mögliche Probleme bei Upgrades von Datenbanken in der Zukunft zu vermeiden, sollten Sie keine Schemanamen verwenden, die mit der Zeichenfolge SYS beginnen. Der Datenbankmanager lässt die Erstellung von Triggern, benutzerdefinierten Typen oder benutzerdefinierten Funktionen, die einen mit SYS beginnenden Schemanamen verwenden, nicht zu.
- Es wird empfohlen, das Wort SESSION nicht als Schemanamen zu verwenden. Deklarierte temporäre Tabellen müssen durch SESSION qualifiziert werden. Daher kann es vorkommen, dass eine Anwendung eine temporäre Tabelle mit einem Namen deklariert, der mit dem einer persistenten Tabelle identisch ist. In diesem Fall kann die Anwendungslogik zu komplex werden. Vermeiden Sie die Verwendung des Schemas SESSION, außer wenn Sie mit deklarierten temporären Tabellen arbeiten.

Namen von begrenzten Bezeichnern und Objekten

Schlüsselwörter dürfen verwendet werden. Wird ein Schlüsselwort in einem Kontext verwendet, in dem es auch als SQL-Schlüsselwort interpretiert werden kann, muss es als begrenzter Bezeichner angegeben werden.

Mithilfe der begrenzten Bezeichner ist es möglich, ein Objekt zu erstellen, dessen Name gegen diese Namenskonventionen verstößt. Bei der späteren Verwendung eines solchen Objekts können jedoch Fehler auftreten. Wenn Sie zum Beispiel eine Spalte mit einem Namen erstellt haben, in dem ein Pluszeichen (+) oder ein Minuszeichen (-) vorkommt, und Sie diese Spalte später in einem Index verwenden, treten Probleme auf, wenn Sie versuchen, die Tabelle zu reorganisieren.

Namenskonventionen für Benutzer, Benutzer-IDs und Gruppen

Für Benutzernamen, Benutzer-IDs und Gruppennamen müssen die geltenden Richtlinien beachtet werden.

Tabelle 83. Namenskonventionen für Benutzer, Benutzer-IDs und Gruppen

Objekte	Richtlinien
• Gruppennamen • Benutzernamen • Benutzer-IDs	• Die Längen von Gruppennamen dürfen die in „SQL- und XML-Begrenzungen" (SQL and XML limits) im Handbuch *SQL Reference* aufgeführte Gruppennamenlänge nicht überschreiten. • Benutzer-IDs auf Linux- und UNIX-Betriebssystemen dürfen bis zu 8 Zeichen enthalten. • Benutzernamen unter Windows dürfen bis zu 30 Zeichen enthalten. • Wenn nicht der Authentifizierungstyp CLIENT verwendet wird, werden Clients unter einem anderen als einem 32-Bit-Windows-Betriebssystem, die auf Windows zugreifen und Benutzernamen verwenden, die länger als die in „SQL- und XML-Begrenzungen" (SQL and XML limits) im Handbuch *SQL Reference* aufgeführte Benutzernamenlänge sind, unterstützt, wenn der Benutzername und das Kennwort explizit angegeben werden. • Für Namen und IDs gilt Folgendes: – USERS, ADMINS, GUESTS, PUBLIC, LOCAL oder für SQL reservierte Wörter sind nicht zulässig. – Sie dürfen nicht mit IBM, SQL oder SYS beginnen.

Anmerkung:

1. Für bestimmte Betriebssysteme muss bei Benutzer-IDs und Kennwörtern die Groß-/Kleinschreibung beachtet werden. Informationen hierzu enthält die Dokumentation des Betriebssystems.
2. Die von einer erfolgreichen CONNECT- oder ATTACH-Operation zurückgegebene Berechtigungs-ID wird auf die Länge für Berechtigungsnamen abgeschnitten, die in „SQL- und XML-Begrenzungen" (SQL and XML limits) im Handbuch *SQL Reference* aufgeführt ist. An die Berechtigungs-ID werden drei Punkte (...) angefügt und die SQLWARN-Felder enthalten Warnungen, die auf die Abtrennung der übrigen Zeichen hinweisen.
3. Folgende Leerzeichen werden aus Benutzer-IDs und Kennwörtern entfernt.
4. **Einschränkungen zur Berechtigungs-ID AUTHID:** In DB2 Version 9.5 und höher kann die Berechtigungs-ID eine Größe von 128 Byte haben. Wenn die Berechtigungs-ID jedoch als Benutzer-ID oder Gruppenname des Betriebssystems interpretiert wird, gelten die Einschränkungen des Betriebssystems. Benutzer-IDs auf Linux- und UNIX-Betriebssystemen dürfen bis zu 8 Zeichen enthalten. Benutzer-IDs auf Windows-Betriebssystemen dürfen bis zu 30 Zeichen für Benutzer-IDs und Gruppennamen enthalten. Daher können Sie zwar eine Berechtigungs-ID mit einer Größe von 128-Byte erteilen, jedoch ist es nicht möglich, als Benutzer, der diese Berechtigungs-ID hat, eine Verbindung herzustellen. Wenn Sie ein eigenes Sicherheits-Plug-in schreiben, können Sie die erweiterten Größen für die Berechtigungs-ID verwenden. Zum Beispiel können Sie Ihr Sicherheits-Plug-in mit einer Benutzer-ID mit einer Größe von 30-Byte versehen, und das Plug-in gibt während der Authentifizierung eine 128-Byte-Berechtigungs-ID zurück, mit der Sie eine Verbindung herstellen können.

Benennungsregeln in einer NLS-Umgebung

Der Standardzeichensatz, der für Datenbanknamen verwendet werden kann, besteht aus den großen und kleinen Einzelbytebuchstaben des lateinischen Alphabets (A...Z, a...z), den arabischen Ziffern (0...9) und dem Unterstreichungszeichen (_).

Diese Liste wird noch um drei Sonderzeichen (#, @ und $) erweitert, um Kompatibilität mit den Hostdatenbankprodukten zu gewährleisten. Die Sonderzeichen #, @ und $ sind in einer NLS-Umgebung mit Vorsicht zu verwenden, da sie nicht im unveränderlichen Zeichensatz für NLS-Hosts (EBCDIC) enthalten sind. Je nach verwendeter Codepage können auch Zeichen aus dem erweiterten Zeichensatz verwendet werden. Wenn Sie die Datenbank in einer Umgebung mit mehreren Codepages verwenden, müssen Sie darauf achten, dass alle Codepages die Elemente aus dem erweiterten Zeichensatz unterstützen, die Sie verwenden möchten.

Bei der Benennung von Datenbankobjekten (wie Tabellen und Sichten), Programmkennsätzen, Hostvariablen und Cursorn können auch Elemente aus dem erweiterten Zeichensatz (z. B. Buchstaben mit diakritischen Zeichen) verwendet werden. Welche Zeichen im Einzelnen verfügbar sind, hängt von der verwendeten Codepage ab.

Definition des erweiterten Zeichensatzes für DBCS-Bezeichner: In DBCS-Umgebungen umfasst der erweiterte Zeichensatz alle Zeichen des Standardzeichensatzes plus die folgenden Elemente:

- Alle Doppelbytezeichen in jeder DBCS-Codepage, mit Ausnahme des Doppelbyteleerzeichens, sind gültige Buchstaben.
- Das Doppelbyteleerzeichen ist ein Sonderzeichen.
- Die in jeder Mischcodepage verfügbaren Einzelbytezeichen werden verschiedenen Kategorien zugeordnet:

Kategorie	Gültige Codepunkte in jeder Mischcodepage
Ziffern	x30-39
Buchstaben	x23-24, x40-5A, x61-7A, xA6-DF (A6-DF nur für Codepages 932 und 942)
Sonderzeichen	Alle anderen gültigen Codepunkte für Einzelbytezeichen

Benennungsregeln in einer Unicode-Umgebung

In einer Unicode-Datenbank weisen alle Bezeichner das UTF-8-Mehrbyteformat auf. Daher ist es möglich, in Bezeichnern beliebige UCS-2-Zeichen zu verwenden, sofern die Verwendung eines Zeichens des erweiterten Zeichensatzes (wie beispielsweise eines Zeichens mit Akzent oder eines Mehrbytezeichens) vom DB2-Datenbanksystem zugelassen wird.

Clients können alle Zeichen eingeben, die von ihrer Umgebung unterstützt werden. Alle Zeichen in den Bezeichnern werden vom Datenbankmanager in UTF-8 umgesetzt. Zwei Punkte müssen beim Angeben von Zeichen der Landessprache in Bezeichnern für eine Unicode-Datenbank beachtet werden:

- Jedes Nicht-ASCII-Zeichen erfordert zwei bis vier Byte. Ein Bezeichner mit n Byte kann daher je nach Verhältnis von ASCII- zu anderen Zeichen nur zwischen $n/4$ und n Zeichen enthalten. Wenn Sie nur ein oder zwei Nicht-ASCII-Zeichen (z. B. Zeichen mit Akzent) haben, liegt die Grenze näher an n Zeichen, während für einen Bezeichner, der nur aus Nicht-ASCII-Zeichen besteht (z. B. in Japanisch), maximal $n/4$ bis $n/3$ Zeichen verwendet werden können.

- Wenn Bezeichner von unterschiedlichen Clientumgebungen eingegeben werden sollen, sollten sie mit der gemeinsamen Untermenge von Zeichen definiert werden, die auf diesen Clients verfügbar sind. Wenn z. B. von Lateinisch-1-, arabischen oder japanischen Umgebungen auf eine Unicode-Datenbank zugegriffen werden soll, sollten alle Bezeichner auf ASCII beschränkt werden.

Kapitel 20. Lightweight Directory Access Protocol (LDAP)

LDAP (Lightweight Directory Access Protocol) ist eine Standardmethode zum Zugriff auf Verzeichnisservices. Bei einem Verzeichnisservice handelt es sich um ein Repository mit Ressourceninformationen zu mehreren Systemen und Services innerhalb einer verteilten Umgebung. Er stellt den Client- und Serverzugriff auf diese Ressourcen bereit.

Jede Datenbankserverinstanz veröffentlicht Informationen über ihre Existenz auf einem LDAP-Server und stellt dem LDAP-Verzeichnis Datenbankinformationen zur Verfügung, wenn die Datenbanken erstellt werden. Wenn ein Client eine Verbindung zur Datenbank herstellt, können die Kataloginformationen für den Server aus dem LDAP-Verzeichnis abgerufen werden. Die einzelnen Clients müssen die Kataloginformationen nun nicht mehr lokal auf den verschiedenen Maschinen speichern. Clientanwendungen durchsuchen das LDAP-Verzeichnis nach den erforderlichen Informationen für die Herstellung der Verbindung zur Datenbank.

Es gibt einen Cachingmechanismus, der es ermöglicht, dass der Client den LDAP-Verzeichnisserver nur einmal durchsuchen muss. Wenn die Informationen aus dem LDAP-Verzeichnisserver abgerufen sind, werden sie auf dem lokalen Computer entsprechend den Werten des Konfigurationsparameters **dir_cache** des Datenbankmanagers und der Registrierdatenbankvariablen **DB2LDAPCACHE** gespeichert bzw. im Cache abgelegt. Der Konfigurationsparameter **dir_cache** des Datenbankmanagers dient zum Speichern von Datenbank-, Knoten- und DCS-Verzeichnisdateien in einem Speichercache. Der Verzeichniscache wird von einer Anwendung genutzt, bis die Anwendung geschlossen wird. Die Registrierdatenbankvariable **DB2LDAPCACHE** dient zum Speichern von Datenbank-, Knoten- und DCS-Verzeichnisdateien in einem lokalen Plattencache.

- Bei **DB2LDAPCACHE**=NO und **dir_cache**=NO werden die Informationen immer aus LDAP gelesen.
- Bei **DB2LDAPCACHE**=NO und **dir_cache**=YES werden die Informationen einmal aus dem LDAP gelesen und in den DB2-Cache eingefügt.
- Wenn **DB2LDAPCACHE**=YES definiert oder wenn diese Variable überhaupt nicht definiert ist, werden die Informationen einmal aus dem LDAP gelesen und in den Cache für lokale Datenbank-, Knoten- und DCS-Verzeichnisse gestellt.

Anmerkung: Die Registrierdatenbankvariable **DB2LDAPCACHE** gilt nur für die Datenbank- und Knotenverzeichnisse.

Sicherheitsaspekte in einer LDAP-Umgebung

Vor dem Zugriff auf Informationen im LDAP-Verzeichnis wird eine Anwendung oder ein Benutzer vom LDAP-Server authentifiziert. Die Authentifizierung wird als *Binden* an den LDAP-Server bezeichnet. Für die im LDAP-Verzeichnis gespeicherten Informationen muss eine Zugriffssteuerung angewendet werden, um zu verhindern, dass anonyme Benutzer Informationen löschen, ändern oder hinzufügen.

Die Zugriffssteuerung wird standardmäßig übernommen und kann auf Containerebene angewendet werden. Wenn ein neues Objekt erstellt wird, übernimmt es dasselbe Sicherheitsattribut wie das übergeordnete Objekt. Ein für den LDAP-Server verfügbares Verwaltungstool kann zum Definieren der Zugriffssteuerung für das Containerobjekt verwendet werden.

Standardmäßig wird die Zugriffssteuerung wie folgt definiert:

- Auf Datenbank- und Knoteneinträge in LDAP hat jeder (auch jeder anonyme) Benutzer Lesezugriff. Nur der Verzeichnisadministrator und der Eigner oder Ersteller des Objekts haben Schreib-/Lesezugriff.
- Auf Benutzerprofile haben der Profileigner und der Verzeichnisadministrator Schreib-/Lesezugriff. Ein Benutzer kann nicht auf das Profil eines anderen Benutzers zugreifen, wenn er keine Verzeichnisadministratorberechtigung hat.

Anmerkung: Die Berechtigungsprüfung wird immer vom LDAP-Server und nicht von DB2 durchgeführt. Die LDAP-Berechtigungsprüfung ist unabhängig von der DB2-Berechtigung. Ein Konto oder eine Berechtigungs-ID mit der Berechtigung SYSADM hat möglicherweise keinen Zugriff auf das LDAP-Verzeichnis.

Wird bei der Ausführung der LDAP-Befehle oder -APIs kein registrierter Bindenamen (bindDN) und kein Kennwort angegeben, führt DB2 eine Bindeoperation mit dem LDAP-Server durch und verwendet dabei die Standardberechtigungsnachweise, denen möglicherweise nicht ausreichende Berechtigungen zur Ausführung der angeforderten Befehle zugeordnet wurden. In diesem Fall wird ein Fehler zurückgegeben.

Sie können den registrierten Bindenamen (bindDN) und das Kennwort eines Benutzers mit den Klauseln USER und PASSWORD der DB2-Befehle und -APIs explizit angeben.

Von DB2 verwendete LDAP-Objektklassen und -Attribute

In den folgenden Tabellen werden die vom DB2-Datenbankmanager verwendeten Objektklassen beschrieben:

Tabelle 84. cimManagedElement

Objektklasse	**cimManagedElement**
Active Directory-LDAP-Anzeigename	Nicht verfügbar
Allgemeiner Active Directory-Name (cn)	Nicht verfügbar
Beschreibung	Stellt eine Basisklasse für viele der Systemverwaltungsobjektklassen im IBM Schema bereit.
Unterklasse von (SubClassOf)	top
Erforderliche(s) Attribut(e)	
Optionale(s) Attribut(e)	description
Typ	abstract
OID (Objekt-ID)	1.3.18.0.2.6.132
GUID (globale eindeutige Kennung)	b3afd63f-5c5b-11d3-b818-002035559151

Tabelle 85. cimSetting

Objektklasse	**cimSetting**
Active Directory-LDAP-Anzeigename	Nicht verfügbar
Allgemeiner Active Directory-Name (cn)	Nicht verfügbar
Beschreibung	Stellt eine Basisklasse für die Konfiguration und Einstellungen im IBM Schema zur Verfügung.
Unterklasse von (SubClassOf)	cimManagedElement
Erforderliche(s) Attribut(e)	

Tabelle 85. cimSetting (Forts.)

Objektklasse	**cimSetting**
Optionale(s) Attribut(e)	settingID
Typ	abstract
OID (Objekt-ID)	1.3.18.0.2.6.131
GUID (globale eindeutige Kennung)	b3afd64d-5c5b-11d3-b818-002035559151

Tabelle 86. eProperty

Objektklasse	**eProperty**
Active Directory-LDAP-Anzeigename	ibm-eProperty
Allgemeiner Active Directory-Name (cn)	ibm-eProperty
Beschreibung	Wird zur Angabe von anwendungsspezifischen Einstellungen für Benutzervorgabeneigenschaften verwendet.
Unterklasse von (SubClassOf)	cimSetting
Erforderliche(s) Attribut(e)	
Optionale(s) Attribut(e)	propertyType cisPropertyType cisProperty cesPropertyType cesProperty binPropertyType binProperty
Typ	structural
OID (Objekt-ID)	1.3.18.0.2.6.90
GUID (globale eindeutige Kennung)	b3afd69c-5c5b-11d3-b818-002035559151

Tabelle 87. DB2Node

Objektklasse	**DB2Node**
Active Directory-LDAP-Anzeigename	ibm-db2Node
Allgemeiner Active Directory-Name (cn)	ibm-db2Node
Beschreibung	Stellt einen DB2-Server dar.
Unterklasse von (SubClassOf)	eSap / ServiceConnectionPoint
Erforderliche(s) Attribut(e)	db2nodeName
Optionale(s) Attribut(e)	db2nodeAlias db2instanceName db2Type host / dNSHostName (siehe Anm. 2) protocolInformation/ServiceBindingInformation
Typ	structural
OID (Objekt-ID)	1.3.18.0.2.6.116

Tabelle 87. DB2Node (Forts.)

Objektklasse	**DB2Node**
GUID (globale eindeutige Kennung)	b3afd65a-5c5b-11d3-b818-002035559151
Besondere Hinweise	1. Die Klasse *DB2Node* wird von der Objektklasse *eSap* unter IBM Tivoli Directory Server und von der Objektklasse *ServiceConnectionPoint* unter Microsoft Active Directory abgeleitet. 2. Das Attribut *host* wird in der IBM Tivoli Directory Server-Umgebung verwendet. Das Attribut *dNSHostName* wird unter Microsoft Active Directory verwendet. 3. Das Attribut *protocolInformation* wird nur in der IBM Tivoli Directory Server-Umgebung verwendet. Im Microsoft Active Directory wird das Attribut *ServiceBindingInformation*, das von der Klasse 'ServiceConnectionPoint' übernommen wird, zur Speicherung der Protokollinformationen verwendet.

Das Attribut *protocolInformation* (in IBM Tivoli Directory Server) oder *ServiceBindingInformation* (in Microsoft Active Directory) im Objekt *DB2Node* enthält die Informationen zum Kommunikationsprotokoll für das Binden des DB2-Datenbankservers. Es besteht aus Token, die das unterstützte Netzprotokoll beschreiben. Die Token werden jeweils durch ein Semikolon getrennt. Zwischen den Token steht kein Leerzeichen. Zur Angabe eines optionalen Parameters kann ein Stern (*) verwendet werden.

Es gibt folgende Token für TCP/IP:

- „TCPIP"
- Server-Hostname oder IP-Adresse
- Servicename (svcename) oder Portnummer (z. B. 50000)
- (Optional) Sicherheit („NONE" oder „SOCKS")

Es gibt folgende Token für benannte Pipes:

- „NPIPE"
- Computername des Servers
- Instanzname des Servers

Tabelle 88. DB2Database

Objektklasse	**DB2Database**
Active Directory-LDAP-Anzeigename	ibm-db2Database
Allgemeiner Active Directory-Name (cn)	ibm-db2Database
Beschreibung	Stellt eine DB2-Datenbank dar.
Unterklasse von (SubClassOf)	top
Erforderliche(s) Attribut(e)	db2databaseName db2nodePtr

Tabelle 88. DB2Database (Forts.)

Objektklasse	**DB2Database**
Optionale(s) Attribut(e)	db2databaseAlias db2additionalParameters db2ARLibrary db2authenticationLocation db2gwPtr db2databaseRelease DCEPrincipalName db2altgwPtr db2altnodePtr
Typ	structural
OID (Objekt-ID)	1.3.18.0.2.6.117
GUID (globale eindeutige Kennung)	b3afd659-5c5b-11d3-b818-002035559151

Tabelle 89. db2additionalParameters

Attribut	**db2additionalParameters**
Active Directory-LDAP-Anzeigename	ibm-db2AdditionalParameters
Allgemeiner Active Directory-Name (cn)	ibm-db2AdditionalParameters
Beschreibung	Enthält alle zusätzlichen Parameter, die zum Herstellen einer Verbindung zum Hostdatenbankserver verwendet werden.
Syntax	Von Groß-/Kleinschreibung unabhängige Zeichenfolge
Maximale Länge	1024
Werte	Mit einem Wert
OID (Objekt-ID)	1.3.18.0.2.4.426
GUID (globale eindeutige Kennung)	b3afd315-5c5b-11d3-b818-002035559151

Tabelle 90. db2authenticationLocation

Attribut	**db2authenticationLocation**
Active Directory-LDAP-Anzeigename	ibm-db2AuthenticationLocation
Allgemeiner Active Directory-Name (cn)	ibm-db2AuthenticationLocation
Beschreibung	Gibt an, wo die Authentifizierung stattfindet.
Syntax	Von Groß-/Kleinschreibung unabhängige Zeichenfolge
Maximale Länge	64
Werte	Mit einem Wert
OID (Objekt-ID)	1.3.18.0.2.4.425
GUID (globale eindeutige Kennung)	b3afd317-5c5b-11d3-b818-002035559151
Hinweise	Zulässige Werte sind: CLIENT, SERVER, DCS, DCE, KERBEROS, SVRENCRYPT oder DCSENCRYPT

Tabelle 91. db2ARLibrary

Attribut	**db2ARLibrary**
Active Directory-LDAP-Anzeigename	ibm-db2ARLibrary
Allgemeiner Active Directory-Name (cn)	ibm-db2ARLibrary
Beschreibung	Name der Anwendungsrequesterbibliothek
Syntax	Von Groß-/Kleinschreibung unabhängige Zeichenfolge
Maximale Länge	256
Werte	Mit einem Wert
OID (Objekt-ID)	1.3.18.0.2.4.427
GUID (globale eindeutige Kennung)	b3afd316-5c5b-11d3-b818-002035559151

Tabelle 92. db2databaseAlias

Attribut	**db2databaseAlias**
Active Directory-LDAP-Anzeigename	ibm-db2DatabaseAlias
Allgemeiner Active Directory-Name (cn)	ibm-db2DatabaseAlias
Beschreibung	Aliasname(n) der Datenbank
Syntax	Von Groß-/Kleinschreibung unabhängige Zeichenfolge
Maximale Länge	1024
Werte	Mit mehreren Werten
OID (Objekt-ID)	1.3.18.0.2.4.422
GUID (globale eindeutige Kennung)	b3afd318-5c5b-11d3-b818-002035559151

Tabelle 93. db2databaseName

Attribut	**db2databaseName**
Active Directory-LDAP-Anzeigename	ibm-db2DatabaseName
Allgemeiner Active Directory-Name (cn)	ibm-db2DatabaseName
Beschreibung	Datenbankname
Syntax	Von Groß-/Kleinschreibung unabhängige Zeichenfolge
Maximale Länge	1024
Werte	Mit einem Wert
OID (Objekt-ID)	1.3.18.0.2.4.421
GUID (globale eindeutige Kennung)	b3afd319-5c5b-11d3-b818-002035559151

Tabelle 94. db2databaseRelease

Attribut	**db2databaseRelease**
Active Directory-LDAP-Anzeigename	ibm-db2DatabaseRelease
Allgemeiner Active Directory-Name (cn)	ibm-db2DatabaseRelease
Beschreibung	Datenbank-Release-Nummer
Syntax	Von Groß-/Kleinschreibung unabhängige Zeichenfolge
Maximale Länge	64
Werte	Mit einem Wert
OID (Objekt-ID)	1.3.18.0.2.4.429

Tabelle 94. db2databaseRelease (Forts.)

Attribut	**db2databaseRelease**
GUID (globale eindeutige Kennung)	b3afd31a-5c5b-11d3-b818-002035559151

Tabelle 95. db2nodeAlias

Attribut	**db2nodeAlias**
Active Directory-LDAP-Anzeigename	ibm-db2NodeAlias
Allgemeiner Active Directory-Name (cn)	ibm-db2NodeAlias
Beschreibung	Aliasname(n) der Knoten
Syntax	Von Groß-/Kleinschreibung unabhängige Zeichenfolge
Maximale Länge	1024
Werte	Mit mehreren Werten
OID (Objekt-ID)	1.3.18.0.2.4.420
GUID (globale eindeutige Kennung)	b3afd31d-5c5b-11d3-b818-002035559151

Tabelle 96. db2nodeName

Attribut	**db2nodeName**
Active Directory-LDAP-Anzeigename	ibm-db2NodeName
Allgemeiner Active Directory-Name (cn)	ibm-db2NodeName
Beschreibung	Knotenname
Syntax	Von Groß-/Kleinschreibung unabhängige Zeichenfolge
Maximale Länge	64
Werte	Mit einem Wert
OID (Objekt-ID)	1.3.18.0.2.4.419
GUID (globale eindeutige Kennung)	b3afd31e-5c5b-11d3-b818-002035559151

Tabelle 97. db2nodePtr

Attribut	**db2nodePtr**
Active Directory-LDAP-Anzeigename	ibm-db2NodePtr
Allgemeiner Active Directory-Name (cn)	ibm-db2NodePtr
Beschreibung	Zeiger auf das Knotenobjekt (DB2Node) für den Datenbankserver, der Eigner der Datenbank ist
Syntax	Definierter Name (DN)
Maximale Länge	1000
Werte	Mit einem Wert
OID (Objekt-ID)	1.3.18.0.2.4.423
GUID (globale eindeutige Kennung)	b3afd31f-5c5b-11d3-b818-002035559151
Besondere Hinweise	Diese Abhängigkeit erlaubt es dem Client, die erforderlichen Informationen zum Übertragungsprotokoll abzurufen, um eine Verbindung zur Datenbank herzustellen.

Tabelle 98. db2altnodePtr

Attribut	**db2altnodePtr**
Active Directory-LDAP-Anzeigename	ibm-db2AltNodePtr
Allgemeiner Active Directory-Name (cn)	ibm-db2AltNodePtr
Beschreibung	Zeiger auf das Knotenobjekt (DB2Node), das den alternativen Datenbankserver darstellt
Syntax	Definierter Name (DN)
Maximale Länge	1000
Werte	Mit mehreren Werten
OID (Objekt-ID)	1.3.18.0.2.4.3093
GUID (globale eindeutige Kennung)	5694e266-2059-4e32-971e-0778909e0e72

Tabelle 99. db2gwPtr

Attribut	**db2gwPtr**
Active Directory-LDAP-Anzeigename	ibm-db2GwPtr
Allgemeiner Active Directory-Name (cn)	ibm-db2GwPtr
Beschreibung	Zeiger auf das Knotenobjekt für den Gateway-Server, über den auf die Datenbank zugegriffen werden kann
Syntax	Definierter Name (DN)
Maximale Länge	1000
Werte	Mit einem Wert
OID (Objekt-ID)	1.3.18.0.2.4.424
GUID (globale eindeutige Kennung)	b3afd31b-5c5b-11d3-b818-002035559151

Tabelle 100. db2altgwPtr

Attribut	**db2altgwPtr**
Active Directory-LDAP-Anzeigename	ibm-db2AltGwPtr
Allgemeiner Active Directory-Name (cn)	ibm-db2AltGwPtr
Beschreibung	Zeiger auf das Knotenobjekt, das den alternativen Gateway-Server darstellt
Syntax	Definierter Name (DN)
Maximale Länge	1000
Werte	Mit mehreren Werten
OID (Objekt-ID)	1.3.18.0.2.4.3092
GUID (globale eindeutige Kennung)	70ab425d-65cc-4d7f-91d8-084888b3a6db

Tabelle 101. db2instanceName

Attribut	**db2instanceName**
Active Directory-LDAP-Anzeigename	ibm-db2InstanceName
Allgemeiner Active Directory-Name (cn)	ibm-db2InstanceName
Beschreibung	Name der Datenbankserverinstanz
Syntax	Von Groß-/Kleinschreibung unabhängige Zeichenfolge
Maximale Länge	256

Tabelle 101. db2instanceName (Forts.)

Attribut	**db2instanceName**
Werte	Mit einem Wert
OID (Objekt-ID)	1.3.18.0.2.4.428
GUID (globale eindeutige Kennung)	b3afd31c-5c5b-11d3-b818-002035559151

Tabelle 102. db2Type

Attribut	**db2Type**
Active Directory-LDAP-Anzeigename	ibm-db2Type
Allgemeiner Active Directory-Name (cn)	ibm-db2Type
Beschreibung	Datenbankservertyp
Syntax	Von Groß-/Kleinschreibung unabhängige Zeichenfolge
Maximale Länge	64
Werte	Mit einem Wert
OID (Objekt-ID)	1.3.18.0.2.4.418
GUID (globale eindeutige Kennung)	b3afd320-5c5b-11d3-b818-002035559151
Hinweise	Zulässige Datenbankservertypen: SERVER, MPP und DCS

Tabelle 103. DCEPrincipalName

Attribut	**DCEPrincipalName**
Active Directory-LDAP-Anzeigename	ibm-DCEPrincipalName
Allgemeiner Active Directory-Name (cn)	ibm-DCEPrincipalName
Beschreibung	DCE-Principal-Name
Syntax	Von Groß-/Kleinschreibung unabhängige Zeichenfolge
Maximale Länge	2048
Werte	Mit einem Wert
OID (Objekt-ID)	1.3.18.0.2.4.443
GUID (globale eindeutige Kennung)	b3afd32d-5c5b-11d3-b818-002035559151

Tabelle 104. cesProperty

Attribut	**cesProperty**
Active Directory-LDAP-Anzeigename	ibm-cesProperty
Allgemeiner Active Directory-Name (cn)	ibm-cesProperty
Beschreibung	Werte dieses Attributs können zur Bereitstellung anwendungsspezifischer Vorgabekonfigurationsparameter verwendet werden. Ein Wert kann zum Beispiel Daten im XML-Format enthalten. Alle Werte dieses Attributs müssen einen einheitlichen Attributwert für 'cesPropertyType' aufweisen.
Syntax	Zeichenfolge mit genauer Beachtung der Groß-/Kleinschreibung
Maximale Länge	32700
Werte	Mit mehreren Werten

Tabelle 104. cesProperty (Forts.)

Attribut	**cesProperty**
OID (Objekt-ID)	1.3.18.0.2.4.307
GUID (globale eindeutige Kennung)	b3afd2d5-5c5b-11d3-b818-002035559151

Tabelle 105. cesPropertyType

Attribut	**cesPropertyType**
Active Directory-LDAP-Anzeigename	ibm-cesPropertyType
Allgemeiner Active Directory-Name (cn)	ibm-cesPropertyType
Beschreibung	Werte dieses Attributs können zum Beschreiben der Syntax, Semantik oder anderer Merkmale aller Werte des Attributs 'cesProperty' verwendet werden. Der Wert „XML" kann zum Beispiel verwendet werden, um anzugeben, dass alle Werte des Attributs 'cesProperty' in der XML-Syntax codiert sind.
Syntax	Von Groß-/Kleinschreibung unabhängige Zeichenfolge
Maximale Länge	128
Werte	Mit mehreren Werten
OID (Objekt-ID)	1.3.18.0.2.4.308
GUID (globale eindeutige Kennung)	b3afd2d6-5c5b-11d3-b818-002035559151

Tabelle 106. cisProperty

Attribut	**cisProperty**
Active Directory-LDAP-Anzeigename	ibm-cisProperty
Allgemeiner Active Directory-Name (cn)	ibm-cisProperty
Beschreibung	Werte dieses Attributs können zur Bereitstellung anwendungsspezifischer Vorgabekonfigurationsparameter verwendet werden. Ein Wert kann zum Beispiel eine INI-Datei enthalten. Alle Werte dieses Attributs müssen einen einheitlichen Attributwert für 'cisPropertyType' aufweisen.
Syntax	Von Groß-/Kleinschreibung unabhängige Zeichenfolge
Maximale Länge	32700
Werte	Mit mehreren Werten
OID (Objekt-ID)	1.3.18.0.2.4.309
GUID (globale eindeutige Kennung)	b3afd2e0-5c5b-11d3-b818-002035559151

Tabelle 107. cisPropertyType

Attribut	**cisPropertyType**
Active Directory-LDAP-Anzeigename	ibm-cisPropertyType
Allgemeiner Active Directory-Name (cn)	ibm-cisPropertyType
Beschreibung	Werte dieses Attributs können zum Beschreiben der Syntax, Semantik oder anderer Merkmale aller Werte des Attributs 'cisProperty' verwendet werden. Der Wert „INI File" kann zum Beispiel verwendet werden, um anzugeben, dass alle Werte des Attributs 'cisProperty' INI-Dateien sind.

Tabelle 107. cisPropertyType (Forts.)

Attribut	**cisPropertyType**
Syntax	Von Groß-/Kleinschreibung unabhängige Zeichenfolge
Maximale Länge	128
Werte	Mit mehreren Werten
OID (Objekt-ID)	1.3.18.0.2.4.310
GUID (globale eindeutige Kennung)	b3afd2e1-5c5b-11d3-b818-002035559151

Tabelle 108. binProperty

Attribut	**binProperty**
Active Directory-LDAP-Anzeigename	ibm-binProperty
Allgemeiner Active Directory-Name (cn)	ibm-binProperty
Beschreibung	Werte dieses Attributs können zur Bereitstellung anwendungsspezifischer Vorgabekonfigurationsparameter verwendet werden. Ein Wert kann z. B. eine Gruppe binär codierter Lotus 1-2-3-Eigenschaften enthalten. Alle Werte dieses Attributs müssen einen einheitlichen Attributwert für 'binPropertyType' aufweisen.
Syntax	binär
Maximale Länge	250000
Werte	Mit mehreren Werten
OID (Objekt-ID)	1.3.18.0.2.4.305
GUID (globale eindeutige Kennung)	b3afd2ba-5c5b-11d3-b818-002035559151

Tabelle 109. binPropertyType

Attribut	**binPropertyType**
Active Directory-LDAP-Anzeigename	ibm-binPropertyType
Allgemeiner Active Directory-Name (cn)	ibm-binPropertyType
Beschreibung	Werte dieses Attributs können zum Beschreiben der Syntax, Semantik oder anderer Merkmale aller Werte des Attributs 'binProperty' verwendet werden. Der Wert „Lotus 123" kann zum Beispiel verwendet werden, um anzugeben, dass alle Werte des Attributs 'binProperty' binär codierte Lotus 123-Eigenschaften sind.
Syntax	Von Groß-/Kleinschreibung unabhängige Zeichenfolge
Maximale Länge	128
Werte	Mit mehreren Werten
OID (Objekt-ID)	1.3.18.0.2.4.306
GUID (globale eindeutige Kennung)	b3afd2bb-5c5b-11d3-b818-002035559151

Tabelle 110. PropertyType

Attribut	**PropertyType**
Active Directory-LDAP-Anzeigename	ibm-propertyType
Allgemeiner Active Directory-Name (cn)	ibm-propertyType
Beschreibung	Werte dieses Attributs beschreiben die semantischen Merkmale des eProperty-Objekts.

Tabelle 110. PropertyType (Forts.)

Attribut	PropertyType
Syntax	Von Groß-/Kleinschreibung unabhängige Zeichenfolge
Maximale Länge	128
Werte	Mit mehreren Werten
OID (Objekt-ID)	1.3.18.0.2.4.320
GUID (globale eindeutige Kennung)	b3afd4ed-5c5b-11d3-b818-002035559151

Tabelle 111. settingID

Attribut	settingID
Active Directory-LDAP-Anzeigename	Nicht verfügbar
Allgemeiner Active Directory-Name (cn)	Nicht verfügbar
Beschreibung	Ein Benennungsattribut, das zum Identifizieren der von 'cimSetting' abgeleiteten Objekteinträge, zum Beispiel 'eProperty', dient.
Syntax	Von Groß-/Kleinschreibung unabhängige Zeichenfolge
Maximale Länge	256
Werte	Mit einem Wert
OID (Objekt-ID)	1.3.18.0.2.4.325
GUID (globale eindeutige Kennung)	b3afd596-5c5b-11d3-b818-002035559151

Erweitern des LDAP-Verzeichnisschemas mit DB2-Objektklassen und -Attributen

Das LDAP-Verzeichnisschema definiert Objektklassen und Attribute für die in den LDAP-Verzeichniseinträgen gespeicherten Informationen. Eine Objektklasse besteht aus einer Reihe von verbindlichen und optionalen Attributen. Jedem Eintrag im LDAP-Verzeichnis ist eine Objektklasse zugeordnet.

Bevor der DB2-Datenbankmanager Informationen in LDAP speichern kann, muss das Verzeichnisschema für den LDAP-Server die vom DB2-Datenbanksystem verwendeten Objektklassen und Attribute enthalten. Das Hinzufügen neuer Objektklassen und Attribute zum Basisschema wird als *Schemaerweiterung* bezeichnet.

Unterstützte LDAP-Client- und -Serverkonfigurationen

Die folgende Tabelle bietet eine Übersicht über die unterstützten LDAP-Client- und -Serverkonfigurationen.

IBM Tivoli Directory Server ist ein LDAP-Server der Version 6.2 und für Windows, AIX, Solaris, Linux und HP-UX verfügbar. Er wird als Teil des Basisbetriebssystems für AIX und System i sowie mit OS/390 Security Server geliefert.

Die DB2-Datenbank unterstützt IBM LDAP Client unter AIX, Solaris, HP-UX 11.11, Windows und Linux.

Microsoft Active Directory-Server ist ein LDAP-Server der Version 3 und als Teil der Betriebssystemfamilien Windows 2000 Server und Windows Server 2003 verfügbar.

Der LDAP-Client von Microsoft ist in das Windows-Betriebssystem integriert.

Tabelle 112. Unterstützte LDAP-Client- und -Serverkonfigurationen

Unterstützte LDAP-Client- und -Server-konfigurationen	IBM Tivoli Directory Server	Microsoft Active Directory-Server	Sun One LDAP-Server
IBM LDAP Client	Unterstützt	Unterstützt	Unterstützt
Microsoft-LDAP/ ADSI- Client	Unterstützt	Unterstützt	Unterstützt

Anmerkung: Bei der Ausführung unter Windows-Betriebssystemen unterstützt der DB2-Datenbankmanager die Verwendung des IBM LDAP-Clients oder des Microsoft-LDAP-Clients. Zur expliziten Auswahl des IBM LDAP-Clients setzen Sie mit dem Befehl **db2set** die Registrierdatenbankvariable DB2LDAP_CLIENT_PROVIDER auf den Wert „IBM". Der LDAP-Client von Microsoft ist in das Windows-Betriebssystem integriert.

LDAP-Unterstützung und DB2 Connect

Wenn die LDAP-Unterstützung auf dem DB2 Connect-Gateway zur Verfügung steht und die Datenbank im Datenbankverzeichnis auf dem Gateway nicht gefunden werden kann, sucht der DB2-Datenbankmanager die Datenbankposition in LDAP und versucht, die gefundenen Informationen zu speichern.

Registrieren von Hostdatenbanken in LDAP

Wenn Sie Hostdatenbanken in LDAP registrieren, sind zwei Konfigurationen möglich: eine Direktverbindung zu den Hostdatenbanken oder eine Verbindung zur Hostdatenbank über ein Gateway.

Informationen zu diesem Vorgang

Für eine Direktverbindung zu den Hostdatenbanken registrieren Sie den Host-Server in LDAP und katalogisieren anschließend die Hostdatenbank in LDAP, indem Sie den Knotennamen des Host-Servers angeben. Für eine Verbindung zur Hostdatenbank über ein Gateway registrieren Sie den Gateway-Server in LDAP und katalogisieren anschließend die Hostdatenbank in LDAP, indem Sie den Knotennamen des Gateway-Servers angeben.

Wenn die LDAP-Unterstützung auf dem DB2 Connect-Gateway zur Verfügung steht und die Datenbank im Datenbankverzeichnis auf dem Gateway nicht gefunden wird, sucht das DB2-Datenbanksystem in LDAP und versucht, die gefundenen Informationen zu speichern.

Vorgehensweise

Sie können Angaben zur Hostdatenbank in LDAP normalerweise manuell konfigurieren, sodass die Clients die Datenbank und den Knoten nicht lokal auf jedem Computer katalogisieren müssen. Gehen Sie wie folgt vor:

1. Registrieren Sie den Hostdatenbankserver in LDAP.

   Dazu müssen Sie den Namen des fernen Computers, den Instanznamen und den Knotentyp für den Hostdatenbankserver im Befehl **REGISTER** angeben, indem Sie die Klauseln **REMOTE**, **INSTANCE** und **NODETYPE** verwenden. Die Klausel **REMOTE** kann entweder auf den Hostnamen oder den LU-Namen der Hostservermaschine gesetzt werden. Die Klausel **INSTANCE** kann auf eine beliebige Zeichenfolge von höchsten acht Zeichen gesetzt werden. (Der Instanzname kann

z. B. auf DB2 gesetzt werden.) Die Klausel **NODETYPE** muss auf den Wert DCS gesetzt werden, um anzugeben, dass es sich um einen Hostdatenbankserver handelt.

2. Registrieren Sie die Hostdatenbank in LDAP mit dem Befehl **CATALOG LDAP DATABASE**.

   Weitere DRDA-Parameter können mit der Klausel **PARMS** angegeben werden. Der Datenbankauthentifizierungstyp sollte auf SERVER gesetzt werden.

Beispiel

Das folgende Beispiel zeigt eine Direktverbindung zu den Hostdatenbanken sowie eine Verbindung zur Hostdatenbank über ein Gateway. Es gibt eine Hostdatenbank mit dem Namen NIAGARA_FALLS, die eingehende Verbindungen über TCP/IP akzeptieren kann. Wenn der Client keine direkte Verbindung zum Host herstellen kann, weil er nicht über DB2 Connect verfügt, stellt er die Verbindung über ein Gateway mit dem Namen goto@niagara her.

Die folgenden Arbeitsschritte müssen ausgeführt werden:

1. Der Hostdatenbankserver muss in LDAP für TCP/IP-Konnektivität registriert werden. Der TCP/IP-Hostname des Servers ist "myhost" und die Portnummer ist "446". Die Klausel **NODETYPE** wird auf den Wert DCS gesetzt, um anzugeben, dass es sich um einen Hostdatenbankserver handelt.

   ```
   db2 register ldap as nftcpip tcpip hostname myhost svcename 446
      remote mvssys instance mvsinst nodetype dcs
   ```

2. Ein DB2 Connect-Gateway-Server muss in LDAP für TCP/IP-Konnektivität registriert werden. Der TCP/IP-Hostname für den Gateway-Server ist "niagara" und die Portnummer ist "50000".

   ```
   db2 register ldap as whasf tcpip hostname niagara svcename 50000
      remote niagara instance goto nodetype server
   ```

3. Die Hostdatenbank muss in LDAP mit TCP/IP-Konnektivität katalogisiert werden. Der Name der Hostdatenbank ist "NIAGARA_FALLS", der Aliasname der Datenbank "nftcpip". Die Klausel **GWNODE** wird verwendet, um den Knotennamen des DB2 Connect-Gateway-Servers anzugeben.

   ```
   db2 catalog ldap database NIAGARA_FALLS as nftcpip at node nftcpip
      gwnode whasf authentication server
   ```

Nach Abschluss der oben gezeigten Registrierung und Katalogisierung können Sie eine Verbindung zu 'nftcpip' herstellen, wenn Sie über TCP/IP auf den Host zugreifen wollen. Wenn auf Ihrer Client-Workstation DB2 Connect nicht installiert ist, wird die Verbindung mit TCP/IP über das Gateway hergestellt. Vom Gateway aus wird die Verbindung zum Host über TCP/IP hergestellt.

Erweitern des Verzeichnisschemas für IBM Tivoli Directory Server

Wenn Sie IBM Tivoli Directory Server verwenden, sind alle für die DB2-Datenbank vor Version 8.2 erforderlichen Objektklassen und Attribute im Basisschema enthalten.

Führen Sie den folgenden Befehl aus, um das Basisschema mit den neuen, seit Version 8.2 eingeführten DB2-Datenbankattributen zu erweitern:

```
ldapmodify -c -h maschinenname:389 -D dn -w kennwort -f altgwnode.ldif
```

Die Datei altgwnode.ldif besitzt folgenden Inhalt:

```
dn:                cn=schema
changetype:        modify
add:               attributetypes
attributetypes:    (
 1.3.18.0.2.4.3092
 NAME 'db2altgwPtr'
 DESC 'DN pointer to DB2 alternate gateway (node) object'
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.12)
-
add:               ibmattributetypes
ibmattributetypes: (
 1.3.18.0.2.4.3092
 DBNAME ('db2altgwPtr' 'db2altgwPtr')
 ACCESS-CLASS NORMAL
 LENGTH 1000)


dn:                cn=schema
changetype:        modify
add:               attributetypes
attributetypes:    (
 1.3.18.0.2.4.3093
 NAME 'db2altnodePtr'
 DESC 'DN pointer to DB2 alternate node object'
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.12)
-
add:               ibmattributetypes
ibmattributetypes: (
 1.3.18.0.2.4.3093
 DBNAME ('db2altnodePtr' 'db2altnodePtr')
 ACCESS-CLASS NORMAL
 LENGTH 1000)


dn:            cn=schema
changetype:    modify
replace:       objectclasses
objectclasses: (
 1.3.18.0.2.6.117
 NAME 'DB2Database'
 DESC 'DB2 database'
 SUP cimSetting
 MUST ( db2databaseName $ db2nodePtr )
 MAY ( db2additionalParameters $ db2altgwPtr $ db2altnodePtr
    $ db2ARLibrary $ db2authenticationLocation $ db2databaseAlias
    $ db2databaseRelease $ db2gwPtr $ DCEPrincipalName ) )
```

Die Dateien `altgwnode.ldif` und `altgwnode.readme` stehen unter folgender URL-Adresse zur zur Verfügung: ftp://ftp.software.ibm.com/ps/products/db2/tools/ldap

Nach dem Hinzufügen der DB2-Schemadefinition muss der Directory Server erneut gestartet werden, um alle Änderungen in Kraft zu setzen.

Netscape-Unterstützung und Attributdefinitionen für das LDAP-Verzeichnis

Der unterstützte Stand für Netscape LDAP Server ist Version 4.12 oder eine spätere Version.

In Netscape LDAP Server Version 4.12 (oder später) ermöglicht der Netscape Directory Server Anwendungen, das Schema durch Hinzufügen von Attribut- und Objektklassendefinitionen in den beiden Dateien `slapd.user_oc.conf` und

slapd.user_at.conf zu erweitern. Diese beiden Dateien befinden sich im Verzeichnis <Netscape_installationspfad>\slapd-<maschinenname>\config.

Anmerkung: Wenn Sie Sun One Directory Server 5.0 verwenden, finden Sie im Abschnitt über das Erweitern des Verzeichnisschemas für Sun One Directory Server entsprechende Informationen.

Die DB2-Attribute müssen der Datei slapd.user_at.conf wie folgt hinzugefügt werden:

```
#########################################################################
#
# IBM DB2 Database
# Attributdefinitionen
#
# bin -> binär
# ces -> Zeichenfolge in exakter Groß-/Kleinschreibung
# cis -> Zeichenfolge ohne Unterscheidung der Groß-/Kleinschreibung
# dn -> registrierter Name (Distinguished Name)
#
#########################################################################

attribute binProperty                    1.3.18.0.2.4.305     bin
attribute binPropertyType                1.3.18.0.2.4.306     cis
attribute cesProperty                    1.3.18.0.2.4.307     ces
attribute cesPropertyType                1.3.18.0.2.4.308     cis
attribute cisProperty                    1.3.18.0.2.4.309     cis
attribute cisPropertyType                1.3.18.0.2.4.310     cis
attribute propertyType                   1.3.18.0.2.4.320     cis
attribute systemName                     1.3.18.0.2.4.329     cis
attribute db2nodeName                    1.3.18.0.2.4.419     cis
attribute db2nodeAlias                   1.3.18.0.2.4.420     cis
attribute db2instanceName                1.3.18.0.2.4.428     cis
attribute db2Type                        1.3.18.0.2.4.418     cis
attribute db2databaseName                1.3.18.0.2.4.421     cis
attribute db2databaseAlias               1.3.18.0.2.4.422     cis
attribute db2nodePtr                     1.3.18.0.2.4.423     dn
attribute db2gwPtr                       1.3.18.0.2.4.424     dn
attribute db2additionalParameters        1.3.18.0.2.4.426     cis
attribute db2ARLibrary                   1.3.18.0.2.4.427     cis
attribute db2authenticationLocation      1.3.18.0.2.4.425     cis
attribute db2databaseRelease             1.3.18.0.2.4.429     cis
attribute DCEPrincipalName               1.3.18.0.2.4.443     cis
```

Die DB2-Objektklassen müssen der Datei slapd.user_oc.conf wie folgt hinzugefügt werden:

```
#########################################################################
#
# IBM DB2 Database
# Objektklassendefinitionen
#
#########################################################################

objectclass eProperty
        oid 1.3.18.0.2.6.90
        requires
                objectClass
        allows
                cn,
                propertyType,
                binProperty,
                binPropertyType,
                cesProperty,
                cesPropertyType,
                cisProperty,
```

```
                cisPropertyType

objectclass eApplicationSystem
        oid 1.3.18.0.2.6.84
        requires
                objectClass,
                systemName


objectclass DB2Node
        oid 1.3.18.0.2.6.116
        requires
                objectClass,
                db2nodeName
        allows
                db2nodeAlias,
                host,
                db2instanceName,
                db2Type,
                description,
                protocolInformation

objectclass DB2Database
        oid 1.3.18.0.2.6.117
        requires
                objectClass,
                db2databaseName,
                db2nodePtr
        allows
                db2databaseAlias,
                description,
                db2gwPtr,
                db2additionalParameters,
                db2authenticationLocation,
                DCEPrincipalName,
                db2databaseRelease,
                db2ARLibrary
```

Nach dem Hinzufügen der DB2-Schemadefinition muss der Directory Server erneut gestartet werden, um alle Änderungen in Kraft zu setzen.

Erweitern des Verzeichnisschemas für Sun One Directory Server

Sun One Directory Server wird auch als Netscape- oder iPlanet-Verzeichnisserver bezeichnet.

Zur funktionsfähigen Einrichtung von Sun One Directory Server in Ihrer Umgebung fügen Sie die Datei `60ibmdb2.ldif` dem folgenden Verzeichnis hinzu:

Unter Windows, wenn iPlanet im Verzeichnis `C:\iPlanet\Servers` installiert ist, fügen Sie die oben genannte Datei dem Verzeichnis `.\sldap-<maschinenname>\config\schema` hinzu.

Unter UNIX, wenn iPlanet im Verzeichnis `/usr/iplanet/servers` installiert ist, fügen Sie die oben genannte Datei dem Verzeichnis `./slapd-<maschinenname>/config/schema` hinzu.

Die Datei besitzt folgenden Inhalt:

```
########################################################################
# IBM DB2 Database
########################################################################
dn: cn=schema
```

```
#############################################################################
# Attributdefinitionen (vor Version 8.2)
#############################################################################
attributetypes: ( 1.3.18.0.2.4.305  NAME 'binProperty'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.5              X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.306  NAME 'binPropertyType'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15             X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.307  NAME 'cesProperty'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15             X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.308  NAME 'cesPropertyType'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15             X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.309  NAME 'cisProperty'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15             X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.310  NAME 'cisPropertyType'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15             X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.320  NAME 'propertyType'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15             X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.329  NAME 'systemName'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.419  NAME 'db2nodeName'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.420  NAME 'db2nodeAlias'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15             X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.428  NAME 'db2instanceName'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.418  NAME 'db2Type'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.421  NAME 'db2databaseName'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.422  NAME 'db2databaseAlias'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15             X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.426  NAME 'db2additionalParameters'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.427  NAME 'db2ARLibrary'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.425  NAME 'db2authenticationLocation'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.429  NAME 'db2databaseRelease'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.443  NAME 'DCEPrincipalName'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.423  NAME 'db2nodePtr'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.424  NAME 'db2gwPtr'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
#############################################################################
# Attributdefinitionen (Version 8.2 und spätere Versionen)
#############################################################################
attributetypes: ( 1.3.18.0.2.4.3092 NAME 'db2altgwPtr'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.12             X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.3093 NAME 'db2altnodePtr'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.12             X-ORIGIN 'IBM DB2' )
#############################################################################
# Objektklassendefinitionen
# DB2 Database Version 8.2 hat die obigen beiden neuen optionalen Attribute.
#############################################################################
objectClasses: ( 1.3.18.0.2.6.90  NAME 'eProperty'
   SUP top STRUCTURAL MAY ( cn $ propertyType $ binProperty
   $ binPropertyType $ cesProperty $ cesPropertyType $ cisProperty
   $ cisPropertyType ) X-ORIGIN 'IBM DB2' )
objectClasses: ( 1.3.18.0.2.6.84  NAME 'eApplicationSystem'
   SUP top STRUCTURAL MUST systemName
      X-ORIGIN 'IBM DB2' )
objectClasses: ( 1.3.18.0.2.6.116 NAME 'DB2Node'
   SUP top STRUCTURAL MUST db2nodeName MAY ( db2instanceName $ db2nodeAlias
   $ db2Type $ description $ host $ protocolInformation )
   X-ORIGIN 'IBM DB2' )
```

```
objectClasses: ( 1.3.18.0.2.6.117 NAME 'DB2Database'
   SUP top STRUCTURAL MUST (db2databaseName $ db2nodePtr ) MAY
   ( db2additionalParameters $ db2altgwPtr $ db2altnodePtr $ db2ARLibrary
   $ db2authenticationLocation $ db2databaseAlias $ db2databaseRelease
   $ db2gwPtr $ DCEPrincipalName $ description )
   X-ORIGIN 'IBM DB2' )
```

Die Dateien `60ibmdb2.ldif` und `60ibmdb2.readme` stehen unter folgender URL-Adresse zur Verfügung: ftp://ftp.software.ibm.com/ps/products/db2/tools/ldap

Nach dem Hinzufügen der DB2-Schemadefinition muss der Directory Server erneut gestartet werden, um alle Änderungen in Kraft zu setzen.

Windows Active Directory

Die DB2-Datenbankserver werden in Active Directory als ibm_db2Node-Objekte bereitgestellt. Die Objektklasse ibm_db2Node ist eine Unterklasse der Objektklasse ServiceConnectionPoint (SCP).

Jedes ibm_db2Node-Objekt enthält Protokollkonfigurationsdaten, die es Clientanwendungen ermöglichen, eine Verbindung zum DB2-Datenbankserver herzustellen. Beim Erstellen einer neuen Datenbank wird diese in Active Directory als ibm_db2Database-Objekt unter dem ibm_db2Node-Objekt bereitgestellt.

Beim Herstellen einer Verbindung zu einer fernen Datenbank fragt der DB2-Client Active Directory über die LDAP-Schnittstelle nach dem ibm_db2Database-Objekt ab. Die Daten zur Protokollkommunikation zur Herstellung der Verbindung zum Datenbankserver (Bindeinformationen) werden aus dem ibm_db2Node-Objekt abgerufen, unter dem das ibm_db2Database-Objekt erstellt ist.

Eigenschaftenseiten für die ibm_db2Node- und ibm_db2Database-Objekte können über das Snap-in **Active Directory-Benutzer und -Computer** der Managementkonsole (MMC) auf einem Domänencontroller angezeigt werden. Zur Einrichtung der Eigenschaftenseite führen Sie den Befehl **regsrv32** wie folgt aus, um die Eigenschaftenseiten für die DB2-Objekte zu registrieren:

```
regsvr32 %DB2PATH%\bin\db2ads.dll
```

Sie können die Objekte über das Snap-in **Active Directory-Benutzer und -Computer** der Managementkonsole (MMC) auf einem Domänencontroller anzeigen. Wählen Sie zum Aufrufen des Verwaltungstools **Start** > **Programme** > **Verwaltung** > **Active Directory-Benutzer und -Computer** aus.

Anmerkung: Sie müssen im Menü **Ansicht** die Option **Benutzer, Gruppen und Computer als Container** auswählen, um die DB2-Datenbankobjekte unterhalb der Computerobjekte anzuzeigen.

Anmerkung: Wenn das DB2-Datenbanksystem nicht auf dem Domänencontroller installiert ist, können Sie die Eigenschaftenseiten von DB2-Datenbankobjekten dennoch anzeigen, indem Sie die Datei `db2ads.dll` aus dem Verzeichnis `%DB2PATH%\bin` und die Ressourcen-DLL-Datei `db2adsr.dll` aus dem Verzeichnis `%DB2PATH%\msg\locale-name` in ein lokales Verzeichnis auf dem Domänencontroller kopieren. (Das Verzeichnis, in dem Sie diese beiden kopierten Dateien ablegen, muss eines der Verzeichnisse sein, die in der Umgebungsvariablen **PATH** definiert sind.) Anschließend führen Sie den Befehl **regsvr32** in dem lokalen Verzeichnis aus, um die DLL-Datei zu registrieren.

Konfigurieren des DB2-Datenbankmanagers zur Verwendung von Active Directory

Für den Zugriff auf Microsoft Active Directory müssen einige Bedingungen erfüllt sein.

Vorbereitende Schritte

- Die Maschine, auf der das DB2-Datenbanksystem ausgeführt wird, muss zu einer Windows 2000- oder Windows Server 2003-Domäne gehören.
- Der Microsoft-LDAP-Client muss installiert sein. Der Microsoft-LDAP-Client ist Bestandteil der Betriebssysteme Windows 2000, Windows XP und Windows Server 2003.

Vorgehensweise

1. Aktivieren Sie die LDAP-Unterstützung.
2. Melden Sie sich bei der Ausführung des DB2-Datenbanksystems über ein Domänenbenutzerkonto an, um Informationen aus Active Directory zu lesen.

Sicherheitsaspekte für Active Directory

Die DB2-Datenbank- und -Knotenobjekte werden im Active Directory unter dem Computerobjekt der Maschine erstellt, auf der der DB2-Server installiert ist. Zum Registrieren eines Datenbankservers oder zum Katalogisieren einer Datenbank im Active Directory müssen Sie über ausreichende Zugriffsberechtigungen zum Erstellen oder Aktualisieren der Objekte unter dem Computerobjekt verfügen.

Standardmäßig können die unter einem Computerobjekt angelegten Objekte von allen authentifizierten Benutzern gelesen und von Administratoren (d. h. Benutzern, die zur Administrator-, Domänenadministrator- und Unternehmensadministratorgruppe gehören) aktualisiert werden. Um einem bestimmten Benutzer bzw. einer bestimmten Gruppe Zugriffsberechtigungen zu erteilen, verwenden Sie das Snap-in **Active Directory-Benutzer und -Computer** der Microsoft Management Console (MMC) wie folgt:

1. Starten Sie das Verwaltungstool **Active Directory-Benutzer und -Computer**:

   Start > **Programme** > **Verwaltung** > **Active Directory-Benutzer und -Computer**
2. Wählen Sie unter **Ansicht** die Option **Erweiterte Funktionen** aus.
3. Wählen Sie den Container **Computer** aus.
4. Klicken Sie mit der rechten Maustaste das Computerobjekt an, das die Servermaschine darstellt, auf der DB2 installiert ist, und wählen Sie **Eigenschaften** aus.
5. Wählen Sie die Registerkarte **Sicherheitseinstellungen** aus und fügen Sie für die angegebene Gruppe bzw. den angegebenen Benutzer die erforderliche Zugriffsberechtigung hinzu.

Die DB2-Registrierdatenbankvariablen und CLI-Einstellungen auf Benutzerebene werden im Objekt für DB2-Eigenschaften unter dem Benutzerobjekt verwaltet. Um die DB2-Registrierdatenbankvariablen oder CLI-Einstellungen auf Benutzerebene zu definieren, muss ein Benutzer über ausreichende Zugriffsberechtigungen verfügen, um Objekte unter dem Benutzerobjekt zu erstellen.

Standardmäßig verfügen nur Administratoren über die Zugriffsberechtigung zum Erstellen von Objekten unter dem Benutzerobjekt. Um einem Benutzer die Zugriffsberechtigung zum Definieren von DB2-Registrierdatenbankvariablen oder CLI-Einstellungen auf Benutzerebene zu erteilen, verwenden Sie das Snap-in **Active Directory-Benutzer und -Computer** der Management Console (MMC) wie folgt:

1. Starten Sie das Verwaltungstool **Active Directory-Benutzer und -Computer**:
   Start > **Programme** > **Verwaltung** > **Active Directory-Benutzer und -Computer**
2. Wählen Sie das Benutzerobjekt im Container **Benutzer** ('Users') aus.
3. Klicken Sie mit der rechten Maustaste das Benutzerobjekt an und wählen Sie **Eigenschaften** aus.
4. Wählen Sie die Registerkarte **Sicherheitseinstellungen** aus.
5. Fügen Sie den Benutzernamen mithilfe der Schaltfläche **Hinzufügen** zu der Liste hinzu.
6. Erteilen Sie die Zugriffsberechtigungen „Schreiben" und „Alle untergeordneten Objekte erstellen".
7. In den erweiterten Einstellungen (Schaltfläche **Erweitert**) wählen Sie im Feld **Übernehmen für** den Eintrag „Dieses und alle untergeordneten Objekte" aus.
8. Wählen Sie das Kontrollkästchen **Vererbbare übergeordnete Berechtigungen übernehmen** aus.

DB2-Objekte im Active Directory

Der DB2-Datenbankmanager erstellt Objekte im Active Directory an den beiden folgenden Positionen:

1. Die DB2-Datenbank- und -Knotenobjekte werden unter dem Computerobjekt der Maschine erstellt, auf der der DB2-Server installiert ist. Für den DB2-Server, der nicht zur Windows-Domäne gehört, werden die DB2-Datenbank- und -Knotenobjekte im Container „System" erstellt.
2. Die DB2-Registrierdatenbankvariablen und CLI-Einstellungen auf der Benutzerebene werden in den DB2-Eigenschaftsobjekten unter dem Benutzerobjekt gespeichert. Diese Objekte enthalten benutzerspezifische Informationen.

Erweitern des Verzeichnisschemas für Active Directory

Bevor der DB2-Datenbankmanager Informationen im Active Directory speichern kann, muss das Verzeichnisschema so erweitert werden, dass es die neuen Objektklassen und Attribute der DB2-Datenbank enthält. Das Hinzufügen neuer Objektklassen und Attribute zum Verzeichnisschema wird als *Schemaerweiterung* bezeichnet.

Informationen zu diesem Vorgang

Sie müssen das Schema für Active Directory durch Ausführung des DB2-Schemainstallationsprogramms **db2schex** erweitern. Führen Sie diesen Befehl vor der Installation der DB2-Produkte und der Erstellung der Datenbanken aus, da Sie andernfalls die Registrierung des Knotens und die Katalogisierung der Datenbanken manuell durchführen müssen.

Das Programm **db2schex** befindet sich auf der Produkt-CD-ROM in folgendem Verzeichnis: `x:\db2\windows\utilities\`. Dabei steht x: für das DVD-Laufwerk.

Zum Aktualisieren des Schemas müssen Sie Mitglied der Schemaadministratorgruppe sein oder über die delegierten Berechtigungen zum Aktualisieren des Schemas verfügen. Führen Sie den folgenden Befehl auf jeder Maschine aus, die Teil der Windows-Domäne ist:

```
runas /user:MyDomain\Administrator x:\db2\Windows\utilities\db2schex.exe
```

Hierbei ist `x:` der DVD-Laufwerkbuchstabe.

Wenn Sie den Befehl **db2schex** aus einer früheren Version des DB2-Datenbankverwaltungssystems ausgeführt haben und diesen Befehl erneut für DB2 UDB Version 8.2 oder eine spätere Version ausführen, werden die beiden folgenden optionalen Attribute der Klasse 'ibm-db2Database' hinzugefügt:

```
ibm-db2AltGwPtr
ibm-db2NodePtr
```

Wenn Sie den Befehl **db2schex** aus einer früheren Version des DB2-Datenbankverwaltungssystems unter Windows nicht ausgeführt haben und diesen Befehl für DB2 Version 9.7 oder eine spätere Version ausführen, werden alle Klassen und Attribute für die LDAP-Unterstützung des DB2-Datenbanksystems hinzugefügt.

Für diesen Befehl sind weitere optionale Klauseln verfügbar. Weitere Informationen finden Sie im Thema „db2schex - Active Directory-Schemaerweiterung (Befehl)".

Beispiele:

- Geben Sie Folgendes ein, um das DB2-Datenbankschema zu installieren:

  ```
  db2schex
  ```

- Geben Sie Folgendes ein, um das DB2-Datenbankschema zu installieren und einen definierten Bindenamen (bindDN) und ein Kennwort anzugeben:

  ```
  db2schex -b "cn=A Name,dc=toronto1,dc=ibm,dc=com"
           -w kennwort
  ```

  Alternativ können Sie auch Folgendes eingeben:

  ```
  db2schex -b Administrator -w kennwort
  ```

- Geben Sie Folgendes ein, um das DB2-Datenbankschema zu deinstallieren:

  ```
  db2schex -u
  ```

- Geben Sie Folgendes ein, um das DB2-Datenbankschema zu deinstallieren und Fehlermeldungen zu ignorieren:

  ```
  db2schex -u -k
  ```

Aktivieren der LDAP-Unterstützung nach Abschluss der Installation

Bevor Sie LDAP verwenden können, müssen Sie nach Abschluss der Installation des DB2-Datenbankprodukts die LDAP-Unterstützung aktivieren.

Vorgehensweise

Gehen Sie wie folgt vor, um die LDAP-Unterstützung zu aktivieren:

1. Führen Sie auf jeder Maschine, die Teil einer Windows-Domäne ist, die folgenden Schritte aus:
   a. Zur Verwendung von Microsoft Active Directory müssen Sie das Verzeichnisschema erweitern, falls Sie dies nicht bereits vor der Installation des DB2-Datenbankprodukts getan haben. Weitere Informationen hierzu finden Sie im Thema „Erweitern des Verzeichnisschemas für Active Directory".
   b. Installieren Sie die Binärdateien der LDAP-Unterstützung, indem Sie **DB2 Setup** ausführen und die Unterstützung für die LDAP-Verzeichnisnutzung in der angepassten Installation auswählen. **DB2 Setup** setzt die DB2-Registrierdatenbankvariable **DB2_ENABLE_LDAP** automatisch auf YES; diese Einstellung ist für die Aktivierung der LDAP-Unterstützung erforderlich.

c. Optional: Wenn Sie den IBM LDAP-Client anstelle des Microsoft-LDAP-Clients verwenden möchten, müssen Sie die Registrierdatenbankvariable **DB2LDAP_CLIENT_PROVIDER** auf IBM setzen.

2. Führen Sie auf jedem LDAP-Client die folgenden Schritte aus:
   a. Geben Sie den TCP/IP-Hostnamen und wahlweise die Portnummer des LDAP-Servers an, indem Sie den folgenden Befehl ausführen: db2set DB2LDAPHOST=*basisdomänenname*[:portnummer]. Dabei ist *basisdomänenname* der TCP/IP-Hostname und *[:portnummer]* die Portnummer. Wenn Sie keine Portnummer angeben, wird die LDAP-Standardportnummer 389 verwendet. Führen Sie für einen für SSL aktivierten LDAP-Server den folgenden Befehl aus: db2set DB2LDAPHOST=*basisdomänenname*:SSL:636. Dabei ist *basisdomänenname* der TCP/IP-Hostname.

      DB2-Objekte befinden sich unter dem definierten LDAP-Basis-DN (baseDN). Sie können den Basis-DN auf den einzelnen Maschinen konfigurieren, indem Sie den folgenden Befehl ausführen:

      ```
      db2set DB2LDAP_BASEDN=basis-DN
      ```

      Dabei ist *basis-DN* der Name des LDAP-Suffixes, das auf dem LDAP-Server definiert ist.
   b. Optional: Wenn Sie LDAP zum Speichern von benutzerspezifischen DB2-Informationen verwenden möchten, geben Sie den DN und das Kennwort des LDAP-Benutzers ein.
3. Wenn Sie das Verzeichnisschema nach der Installation des DB2-Datenbankprodukts erweitert haben, führen Sie die folgenden Schritte aus:
   a. Registrieren Sie die aktuelle Instanz des DB2-Servers bei LDAP, indem Sie den folgenden Befehl ausführen:

      ```
      db2 register ldap as knotenname protocol tcpip
      ```
   b. Registrieren Sie bestimmte Datenbanken bei LDAP, indem Sie den folgenden Befehl ausführen:

      ```
      db2 catalog ldap database dbname as db_aliasname
      ```

Nächste Schritte

Sie können nun die LDAP-Einträge registrieren.

Anmerkung: Wenn der Parameter **DB2LDAPHOST** auf einen für SSL aktivierten LDAP-Server gesetzt wird, können Kataloginformationen nur über SSL von einem LDAP-Server abgerufen werden. Weitere Informationen dazu, wie sichergestellt wird, dass Daten, die zwischen dem Client und dem Server ausgetauscht werden, mit SSL verschlüsselt werden, finden Sie im Abschnitt zum „Konfigurieren von SSL-Unterstützung im DB2-Client".

Registrieren von LDAP-Einträgen

Registrierung von DB2-Servern nach der Installation

Jede DB2-Serverinstanz muss in LDAP registriert werden, um die Protokollkonfigurationsdaten zu veröffentlichen, die von den Clientanwendung zum Herstellen einer Verbindung zu dieser DB2-Serverinstanz verwendet werden.

Informationen zu diesem Vorgang

Beim Registrieren einer Instanz des Datenbankservers müssen Sie einen *Knotennamen* angeben. Der Knotenname wird von Clientanwendungen verwendet,

wenn sie eine Verbindung zum Server herstellen. Sie können mit dem Befehl **CATALOG LDAP NODE** einen anderen Aliasnamen für den LDAP-Knoten katalogisieren.

Anmerkung: Wenn Sie in einer Domänenumgebung unter Windows arbeiten, wird die DB2-Serverinstanz während der Installation im Active Directory automatisch mit den folgenden Informationen registriert:

```
Knotenname: TCP/IP-Hostname
Protokolltyp: TCP/IP
```

Wenn die Länge des TCP/IP-Hostnamens 8 Zeichen überschreitet, wird der Name auf 8 Zeichen abgeschnitten.

Der Befehl **REGISTER** sieht wie folgt aus:

```
db2 register db2 server in ldap
   as ldap-knotenname
   protocol tcpip
```

Die Klausel protocol gibt das Übertragungsprotokoll an, das zum Herstellen einer Verbindung zu diesem Datenbankserver verwendet wird.

Beim Erstellen einer Instanz für DB2 Enterprise Server Edition, die mehrere physische Maschinen umfasst, muss der Befehl **REGISTER** für jeden Computer einmal aufgerufen werden. Verwenden Sie den Befehl **rah**, um den Befehl **REGISTER** an alle Computer abzusetzen.

Anmerkung: In *ldap-knotenname* kann nicht derselbe Knotenname für alle Computer verwendet werden, da der Name in LDAP eindeutig sein muss. Sie müssen im Befehl **REGISTER** den Hostnamen jedes Computers anstelle von *ldap-knotenname* verwenden. Beispiel:

```
rah ">DB2 REGISTER DB2 SERVER IN LDAP AS <> PROTOCOL TCPIP"
```

"<>" wird durch den Hostnamen jedes Computers ersetzt, auf dem der Befehl **rah** ausgeführt wird. Für den seltenen Fall, dass mehrere Instanzen von DB2 Enterprise Server Edition vorhanden sind, kann die Kombination aus Instanz und Hostindex als Knotenname im Befehl **rah** verwendet werden.

Der Befehl **REGISTER** kann für einen fernen DB2-Server abgesetzt werden. Dazu müssen Sie den fernen Systemnamen, den Instanznamen und die Protokollkonfigurationsparameter beim Registrieren eines fernen Servers angeben. Der Befehl kann folgendermaßen verwendet werden:

```
db2 register db2 server in ldap
   as ldap-knotenname
   protocol tcpip
   hostname hostname
   svcename tcpip-servicename
   remote name_des_fernen_computers
   instance instanzname
```

Für den Computernamen gilt folgende Konvention:

- Wenn TCP/IP konfiguriert ist, muss der Systemname mit dem TCP/IP-Hostnamen übereinstimmen.

Beim Betrieb in einer Umgebung mit hoher Verfügbarkeit oder Funktionsübernahme (Failover) und bei Verwendung von TCP/IP als Übertragungsprotokoll muss die Cluster-IP-Adresse verwendet werden. Mit der Cluster-IP-Adresse kann der Client eine Verbindung zum Server auf einem der Computer herstellen, ohne für

jeden Computer einen separaten TCP/IP-Knoten katalogisieren zu müssen. Die Cluster-IP-Adresse wird mit der Klausel hostname wie folgt angegeben:

```
db2 register db2 server in ldap
   as ldap-knotenname
   protocol tcpip
   hostname n.nn.nn.nn
```

Dabei ist *n.nn.nn.nn* die Cluster-IP-Adresse.

Zur Registrierung des DB2-Servers in LDAP aus einer Clientanwendung heraus rufen Sie die API db2LdapRegister auf.

Katalogisieren eines Knoten-Aliasnamens für ATTACH

Beim Registrieren eines DB2-Servers in LDAP muss ein Knotenname für den Server angegeben werden. Anwendungen verwenden den Knotennamen zum Herstellen einer Verbindung zum Datenbankserver.

Vorgehensweise

- Wenn Sie einen anderen Knotennamen benötigen, zum Beispiel wenn der Knotenname in einer Anwendung fest codiert ist, verwenden Sie den Befehl **CATALOG LDAP NODE**, um die Änderung vorzunehmen. Beispiel:

  ```
  db2 catalog ldap node ldap_knotenname
     as neuer_aliasname
  ```

- Zum Entfernen eines LDAP-Knotens aus dem Katalog verwenden Sie den Befehl **UNCATALOG LDAP NODE**. Beispiel:

  ```
  db2 uncatalog ldap node ldap_knotenname
  ```

Registrierung von Datenbanken im LDAP-Verzeichnis

Bei der Erstellung einer Datenbank in einer Instanz wird die Datenbank automatisch in LDAP registriert. Die Registrierung erlaubt Verbindungen ferner Clients zur Datenbank, ohne dass die Datenbank und der Knoten auf dem Client-Computer katalogisiert werden müssen.

Wenn ein Client versucht, eine Verbindung zu einer Datenbank herzustellen, wird das LDAP-Verzeichnis durchsucht, wenn die Datenbank nicht im Datenbankverzeichnis auf dem lokalen Computer vorhanden ist.

Informationen zu diesem Vorgang

Wenn der Name im LDAP-Verzeichnis vorhanden ist, wird die Datenbank trotzdem auf dem lokalen Computer erstellt, allerdings mit einer Warnung bezüglich der Namensunverträglichkeit im LDAP-Verzeichnis. Aus diesem Grund können Sie eine Datenbank manuell im LDAP-Verzeichnis katalogisieren. Der Benutzer kann Datenbanken auf einem fernen Server in LDAP mit dem Befehl **CATALOG LDAP DATABASE** registrieren. Beim Registrieren einer fernen Datenbank geben Sie den Namen des LDAP-Knotens an, der den fernen Datenbankserver darstellt. Sie müssen den fernen Datenbankserver in LDAP mit dem Befehl **REGISTER DB2 SERVER IN LDAP** registrieren, bevor Sie die Datenbank registrieren.

Vorgehensweise

- Verwenden Sie den Befehl **CATALOG LDAP DATABASE**, um eine Datenbank manuell in LDAP zu registrieren:

  ```
     db2 catalog ldap database dbname
  at node knotenname
        with "Meine LDAP-Datenbank"
  ```

- Zur Registrierung einer Datenbank in LDAP aus einer Clientanwendung heraus rufen Sie die API db2LdapCatalogDatabase auf.

Zurücknehmen registrierter LDAP-Einträge

Zurücknehmen der Registrierung des DB2-Servers

Beim Zurücknehmen der Registrierung einer Instanz in LDAP werden auch alle Knoten- oder Aliasnamenobjekte sowie Datenbankobjekte entfernt, die auf die Instanz verweisen.

Informationen zu diesem Vorgang

Zum Zurücknehmen der Registrierung des DB2-Servers auf einem lokalen oder fernen Computer muss der LDAP-Knotenname für den Server angegeben werden.

Vorgehensweise

Nehmen Sie die Registrierung des DB2-Servers wie folgt bei LDAP zurück:

- Geben Sie den Befehl **DEREGISTER** in die Befehlszeile ein:

```
db2 deregister db2 server in ldap
   node knotenname
```

- Rufen Sie aus einer Clientanwendung heraus die API db2LdapDeregister auf.

Ergebnisse

Wenn die Registrierung des DB2-Servers zurückgenommen wird, werden alle LDAP-Knoteneinträge und LDAP-Datenbankeinträge, die auf dieselbe Instanz im DB2-Server verweisen, ebenfalls aus dem Katalog entfernt.

Zurücknehmen der Registrierung der Datenbank aus dem LDAP-Verzeichnis

Die Registrierung der Datenbank in LDAP wird automatisch zurückgenommen, wenn die Datenbank gelöscht wird oder wenn die Registrierung der Eignerinstanz in LDAP zurückgenommen wird.

Vorgehensweise

Gehen Sie wie folgt vor, um die Registrierung für eine Datenbank im LDAP-Verzeichnis zurückzunehmen:

- Geben Sie den Befehl **UNCATALOG LDAP DATABASE** in die Befehlszeile ein:

```
db2 uncatalog ldap database datenbankname
```

- Rufen Sie aus einer Clientanwendung heraus die API **db2LdapUncatalogDatabase** auf.

Konfigurieren von LDAP-Benutzern

Erstellen eines LDAP-Benutzers

Beim Einsatz des IBM Tivoli-Verzeichnisses müssen Sie einen LDAP-Benutzer definieren, bevor Informationen auf Benutzerebene unter LDAP gespeichert werden können. Sie können einen LDAP-Benutzer durch Erstellen einer LDIF-Datei erstel-

len, in der alle Attribute für das Benutzerobjekt enthalten sind. Führen Sie anschließend das Importdienstprogramm LDIF aus, um das Objekt in das LDAP-Verzeichnis zu importieren.

Informationen zu diesem Vorgang

Das DB2-Datenbanksystem unterstützt das Definieren von DB2-Registrierdatenbankvariablen und die CLI-Konfiguration auf Benutzerebene. (Diese Funktion steht auf Linux- und UNIX-Plattformen nicht zur Verfügung.) Durch die Unterstützung auf Benutzerebene stehen in Mehrplatzsystemumgebungen benutzerspezifische Einstellungen zur Verfügung. Ein Beispiel ist der Windows Terminal Server, bei dem jeder angemeldete Benutzer seine Umgebung anpassen kann, ohne dass sich dadurch Konflikte mit der Systemumgebung oder der Umgebung eines anderen Benutzers ergeben.

Das Dienstprogramm **LDIF** für IBM Tivoli Directory Server hat den Namen **LDIF2DB**.

Eine LDIF-Datei mit den Attributen für einen Personenobjekt sieht etwa wie folgt aus:

```
   File name: newuser.ldif

dn: cn=Mary Burnnet, ou=DB2 Development, ou=Toronto, o=ibm, c=ca
objectclass: ePerson
cn: Mary Burnnet
sn: Burnnet
uid: mburnnet
userPassword: password
telephonenumber: 1-416-123-4567
facsimiletelephonenumber: 1-416-123-4568
title: Software Developer
```

Das folgende Beispiel zeigt den **LDIF**-Befehl zum Importieren einer LDIF-Datei mit dem IBM **LDIF**-Importdienstprogramm:

```
LDIF2DB -i newuser.ldif
```

Anmerkung:

1. Der Befehl **LDIF2DB** muss auf dem LDAP-Server ausgeführt werden.
2. Sie müssen die erforderlichen Zugriffsberechtigungen (ACL) für das LDAP-Benutzerobjekt erteilen, damit der LDAP-Benutzer sein eigenes Objekt hinzufügen, löschen, lesen und schreiben kann. Verwenden Sie das Tool LDAP Directory Server Web Administration, um für das Benutzerobjekt die ACL-Zugriffsberechtigungen zu erteilen.

Konfigurieren des LDAP-Benutzers für DB2-Anwendungen

Bei Verwendung des Microsoft LDAP-Clients stimmt der LDAP-Benutzer mit dem Benutzerkonto des Betriebssystems überein. Wenn Sie den IBM LDAP-Client jedoch verwenden, bevor Sie den DB2-Datenbankmanager verwenden, müssen Sie den definierten LDAP-Benutzernamen (DN) und das Kennwort für den zurzeit angemeldeten Benutzer konfigurieren.

Vorgehensweise

Verwenden Sie zum Konfigurieren des definierten Namens (DN) und des Kennworts des LDAP-Benutzers das Dienstprogramm **db2ldcfg**:

```
db2ldcfg -u benutzer-DN -w kennwort --> DN und Kennwort des Benutzers festlegen
         -r                         --> DN und Kennwort des Benutzers löschen
```

Beispiel:

```
db2ldcfg -u "cn=Mary Burnnet,ou=DB2 Development,ou=Toronto,o=ibm,c=ca"
         -w kennwort
```

Definieren von DB2-Registrierdatenbankvariablen auf Benutzerebene in der LDAP-Umgebung

In der LDAP-Umgebung können die DB2-Variablen der Profilregistrierdatenbank auf Benutzerebene definiert werden. Dadurch können Benutzer ihre eigene DB2-Umgebung anpassen.

Informationen zu diesem Vorgang

DB2 Database for Linux, UNIX and Windows arbeitet mit einem Caching-Verfahren. Die DB2-Variablen der Profilregistrierdatenbank auf Benutzerebene werden auf dem lokalen Computer zwischengespeichert.

Der Cache wird in folgenden Fällen aktualisiert:

- Sie aktualisieren eine DB2-Registrierdatenbankvariable auf Benutzerebene oder setzen sie zurück.
- Sie geben den folgenden Befehl zum Aktualisieren der LDAP-Profilvariablen auf Benutzerebene aus:

  ```
  db2set -ur
  ```

Vorgehensweise

Um die DB2-Variablen der Profilregistrierdatenbank auf Benutzerebene zu definieren, verwenden Sie die Option **-ul**:

```
db2set -ul variable=wert
```

Anmerkung: Diese Funktion wird unter den Betriebssystemen AIX und Solaris nicht unterstützt.
Wenn der Parameter **-ul** angegeben wird, liest DB2 Database for Linux, UNIX and Windows die DB2-Registrierdatenbankvariablen immer aus dem Cache.

Inaktivieren der LDAP-Unterstützung

Vorgehensweise

Gehen Sie wie folgt vor, um die LDAP-Unterstützung zu inaktivieren:

1. Nehmen Sie für jede Instanz des DB2-Servers die Registrierung des DB2-Servers in LDAP zurück:

   ```
   db2 deregister db2 server in ldap node knotenname
   ```

2. Setzen Sie die Variable **DB2_ENABLE_LDAP** der DB2-Profilregistrierdatenbank auf den Wert NO.

Aktualisieren der Protokollinformationen für den DB2-Server

Die DB2-Serverinformationen in LDAP müssen immer aktuell sein. Änderungen an den Protokollkonfigurationsparametern oder der Server-Netzadresse beispielsweise machen eine LDAP-Aktualisierung erforderlich.

Informationen zu diesem Vorgang

Beispiele für Protokollkonfigurationsparameter, die aktualisiert werden können, sind Parameter für TCP/IP-Hostname und Servicename oder Portnummer.

Vorgehensweise

- Aktualisieren Sie den DB2-Server in LDAP auf dem lokalen Computer mit dem Befehl **UPDATE LDAP**.
- Zum Aktualisieren von Protokollkonfigurationsparametern eines fernen DB2-Servers verwenden Sie den Befehl **UPDATE LDAP** mit der Klausel **node**:

```
db2 update ldap
   node knotenname
   hostname hostname
   svcename tcpip-servicename
```

Weiterleiten von LDAP-Clients an andere Server

Die Möglichkeit zur Weiterleitung von Clients bei Systemausfällen steht auch bei Einsatz von LDAP zur Verfügung.

Vorbereitende Schritte

Die Registrierdatenbankvariable DB2_ENABLE_LDAP muss auf den Wert „Yes" sein.

Informationen zu diesem Vorgang

In einer LDAP-Umgebung werden sämtliche Datenbank- und Knotenverzeichnisdaten auf einem LDAP-Server verwaltet. Der Client ruft Daten aus dem LDAP-Verzeichnis ab. Diese Daten werden in den lokalen Datenbank- und Knotenverzeichnissen aktualisiert, wenn die Registrierdatenbankvariable DB2LDAPCACHE auf den Wert „Yes" gesetzt ist.

Verwenden Sie den Befehl **UPDATE ALTERNATE SERVER FOR LDAP DATABASE**, um den alternativen Server für eine Datenbank zu definieren, die die DB2-Datenbank in LDAP darstellt. Alternativ können Sie die API 'db2LdapUpdateAlternateServerForDB' von einer Clientanwendung aus aufrufen, um den alternativen Server für die Datenbank in LDAP zu aktualisieren.

Wenn die Daten über diesen alternativen Server eingerichtet sind, werden sie an den Client zurückgegeben, wenn eine Verbindung hergestellt wird.

Anmerkung: Es wird dringend empfohlen, die im LDAP-Server gespeicherten Daten über den alternativen Server mit den Daten über den alternativen Server, die in der Datenbankserverinstanz gespeichert sind, synchron zu halten. Das Ausführen des Befehls **UPDATE ALTERNATE SERVER FOR DATABASE** (beachten Sie, dass es nicht "FOR LDAP DATABASE" heißt) in der Datenbankserverinstanz hilft bei der Sicherstellung der Synchronisierung zwischen der Datenbankserverinstanz und dem LDAP-Server.

Wenn Sie den Befehl **UPDATE ALTERNATE SERVER FOR DATABASE** in der Serverinstanz eingeben oder wenn die LDAP-Unterstützung auf dem Server aktiviert (DB2_ENABLE_LDAP=Yes) ist und die LDAP-Benutzer-ID und das Kennwort im Cache gespeichert werden (der Befehl **db2ldcfg** wurde zuvor ausgeführt), dann wird der alternative Server für die Datenbank automatisch bzw. implizit auf dem

LDAP-Server aktualisiert. Dies funktioniert genau so, wie bei einer expliziten Eingabe des Befehls UPDATE ALTERNATE SERVER FOR LDAP DATABASE.

Wenn der Befehl **UPDATE ALTERNATE SERVER FOR LDAP DATABASE** von einer anderen Instanz als der Datenbankserverinstanz aus abgesetzt wird, stellen Sie mithilfe des Befehls **UPDATE ALTERNATE SERVER FOR DATABASE** sicher, dass die Daten über den alternativen Server in der Datenbankserverinstanz identisch konfiguriert werden. Nachdem der Client zu Anfang eine Verbindung zur Datenbankserverinstanz hergestellt hat, erhalten die Daten über den alternativen Server, die aus der Datenbankserverinstanz zurückgegeben werden, Vorrang vor den Daten, die im LDAP-Server konfiguriert sind. Wenn die Datenbankserverinstanz keine konfigurierten Daten über einen alternativen Server hat, wird die Clientweiterleitung nach der einleitenden Verbindung als inaktiviert betrachtet.

Herstellen einer ATTACH-Verbindung zu einem fernen Server in der LDAP-Umgebung

In der LDAP-Umgebung können Sie eine Verbindung zu einem fernen Datenbankserver mit dem LDAP-Knotennamen im Befehl **ATTACH** herstellen: `db2 attach to` *`ldap-knotenname`*.

Informationen zu diesem Vorgang

Wenn eine Clientanwendung zum ersten Mal eine Verbindung zu einem Knoten oder einer Datenbank herstellt, durchsucht der Datenbankmanager das LDAP-Verzeichnis nach dem Zielknoteneintrag, weil der Knoten im lokalen Knotenverzeichnis nicht definiert ist. Wenn der Eintrag im LDAP-Verzeichnis gefunden wird, werden die Protokollinformationen zum fernen Server abgerufen. Wenn Sie eine Verbindung zur Datenbank herstellen und der Eintrag im LDAP-Verzeichnis gefunden wird, werden die Datenbankinformationen ebenfalls abgerufen. Mithilfe dieser Informationen katalogisiert der Datenbankmanager automatisch einen Datenbankeintrag und einen Knoteneintrag auf dem lokalen Computer. Wenn die Clientanwendung das nächste Mal eine Verbindung zum selben Knoten oder zur selben Datenbank herstellt, werden die Informationen im lokalen Datenbankverzeichnis verwendet, sodass das LDAP-Verzeichnis nicht durchsucht werden muss.

Es gibt einen Cachingmechanismus, der dafür sorgt, dass der Client den LDAP-Server nur einmal durchsucht. Wenn die Informationen abgerufen sind, werden sie auf dem lokalen Computer nach Maßgabe der Werte des Konfigurationsparameters **dir_cache** des Datenbankmanagers und der Registrierdatenbankvariablen **DB2LDAPCACHE** gespeichert bzw. im Cache abgelegt.

- Bei **DB2LDAPCACHE**=NO und **dir_cache**=NO werden die Informationen immer aus LDAP gelesen.
- Bei **DB2LDAPCACHE**=NO und **dir_cache**=YES werden die Informationen einmal aus dem LDAP gelesen und in den DB2-Cache eingefügt.
- Wenn **DB2LDAPCACHE**=YES definiert oder wenn diese Variable überhaupt nicht definiert ist, werden die Informationen einmal beim LDAP-Server gelesen und in den Cache für lokale Datenbank-, Knoten- und DCS-Verzeichnisse gestellt.

Anmerkung: Das Caching von LDAP-Informationen ist für CLI-Variablen oder Variablen der DB2-Profilregistrierdatenbank auf Benutzerebene nicht gültig.

Aktualisieren der LDAP-Einträge in lokalen Datenbank- und Knotenverzeichnissen

Für das DB2-Datenbanksystem steht ein Caching-Mechanismus zur Verfügung, durch den die Häufigkeit reduziert werden kann, mit der ein Client den LDAP-Server durchsucht.

Informationen zu diesem Vorgang

Wenn die Informationen abgerufen sind, werden sie auf dem lokalen Computer nach Maßgabe der Werte des Konfigurationsparameters **dir_cache** des Datenbankmanagers und der Registrierdatenbankvariablen **DB2LDAPCACHE** gespeichert bzw. im Cache abgelegt.

- Bei **DB2LDAPCACHE**=NO und **dir_cache**=NO werden die Informationen immer aus LDAP gelesen.
- Bei **DB2LDAPCACHE**=NO und **dir_cache**=YES werden die Informationen einmal aus dem LDAP gelesen und in den DB2-Cache eingefügt.
- Wenn **DB2LDAPCACHE**=YES definiert oder wenn diese Variable überhaupt nicht definiert ist, werden die Informationen einmal beim LDAP-Server gelesen und in den Cache für lokale Datenbank-, Knoten- und DCS-Verzeichnisse gestellt.

Anmerkung: Das Caching von LDAP-Informationen ist für CLI-Variablen oder Variablen der DB2-Profilregistrierdatenbank auf Benutzerebene nicht gültig. Da LDAP-Informationen Änderungen unterliegen, kann es erforderlich sein, die LDAP-Einträge, die in den lokalen Datenbank- und Knotenverzeichnissen zwischengespeichert sind, zu aktualisieren. Hierfür gibt es eine Reihe möglicher Vorgehensweisen.

Vorgehensweise

- Verwenden Sie den folgenden Befehl, um alle lokalen Datenbank- und Knoteneinträge zu aktualisieren, die von LDAP abgerufen wurden:

  ```
  db2 refresh ldap immediate
  ```

- Verwenden Sie den folgenden Befehl, um vorhandene lokale Datenbank- und Knoteneinträge zu aktualisieren und neue Einträge vom LDAP-Server hinzuzufügen:

  ```
  db2 refresh ldap immediate all
  ```

  Durch die Angabe der Option **IMMEDIATE ALL** werden alle im LDAP-Server enthaltenen Knoten- und Datenbankeinträge zu den lokalen Verzeichnissen hinzugefügt.

- Alternativ dazu können Sie den folgenden Befehl verwenden, um das Aktualisieren der Datenbankeinträge, die auf LDAP-Ressourcen verweisen, durch DB2 Database for Linux, UNIX and Windows beim nächsten Herstellen der Datenbankverbindung bzw. bei der nächsten Instanzverbindung zu erzwingen:

  ```
  db2 refresh ldap database directory
  ```

- Entsprechend können Sie den folgenden Befehl verwenden, um das Aktualisieren der Knoteneinträge, die auf LDAP-Ressourcen verweisen, durch den DB2-Datenbankmanager beim nächsten Herstellen der Datenbankverbindung bzw. bei der nächsten Instanzverbindung zu erzwingen:

  ```
  db2 refresh ldap node directory
  ```

Ergebnisse

Im Rahmen der Aktualisierung werden alle LDAP-Einträge entfernt, die in den lokalen Datenbank- und Knotenverzeichnissen gespeichert sind. Beim nächsten Zugriff der Anwendung auf die Datenbank oder den Knoten liest sie die Informationen direkt aus LDAP und generiert einen neuen Eintrag im lokalen Datenbank- oder Knotenverzeichnis.

Nächste Schritte

Sie können wie folgt vorgehen, um eine zeitgerechte Aktualisierung sicherzustellen:

- Planen Sie eine in regelmäßigen Abständen durchgeführte Aktualisierung.
- Führen Sie den Befehl **REFRESH** beim Systemstart aus.
- Verwenden Sie ein verfügbares Verwaltungspaket, um den Befehl **REFRESH** auf allen Client-Computern aufzurufen.
- Definieren Sie **DB2LDAPCACHE**=NO, um das Caching von LDAP-Informationen in den Datenbank-, Knoten- und DCS-Verzeichnissen zu verhindern.

Durchsuchen von LDAP-Servern

Das DB2-Datenbanksystem durchsucht den aktuellen LDAP-Server. In einer Umgebung, in der mehrere LDAP-Server vorhanden sind, können Sie den Suchbereich jedoch definieren.

Informationen zu diesem Vorgang

Wenn beispielsweise die Informationen im aktuellen LDAP-Server nicht gefunden werden, können Sie eine automatische Durchsuchung aller anderen LDAP-Server angeben oder, alternativ, den Suchbereich allein auf den aktuellen LDAP-Server oder auf den lokalen DB2-Datenbankkatalog einschränken.

Wenn Sie den Suchbereich festlegen, wird dadurch der Standardsuchbereich für das gesamte Unternehmen festgelegt. Der Suchbereich wird durch die Variable **DB2LDAP_SEARCH_SCOPE** der DB2-Profildatenbank gesteuert. Zum Festlegen des Suchbereichswerts verwenden Sie die Option **-gl** (d. h. global in LDAP) im Befehl **db2set**:

```
db2set -gl db2ldap_search_scope = wert
```

Mögliche Werte sind: `local`, `domain` und `global`. Wenn diese Registrierdatenbankvariable nicht definiert ist, gilt der Standardwert `domain`, der den Suchbereich auf das Verzeichnis auf dem aktuellen LDAP-Server einschränkt.

Sie können zum Beispiel den Suchbereich zunächst auf den Wert `global` setzen, nachdem eine neue Datenbank erstellt wurde. Dadurch kann jeder zur Verwendung von LDAP konfigurierte DB2-Client alle LDAP-Server durchsuchen, um die Datenbank zu finden. Sobald der Eintrag nach der ersten Verbindung (CONNECT oder ATTACH) für jeden Client auf jedem Computer erfasst wurde (sofern Sie das Caching aktiviert haben), kann der Suchbereich in `local` geändert werden. Nachdem der Suchbereich in `local` geändert wurde, durchsuchen die Clients keine LDAP-Server mehr.

Anmerkung: Die Variablen **DB2LDAP_KEEP_CONNECTION** und **DB2LDAP_SEARCH_SCOPE** der DB2-Profildatenbank sind die einzigen Registrierdatenbankvariablen, die auf der globalen Ebene in LDAP definiert werden können.

Kapitel 21. SQL- und XML-Begrenzungen

In den Tabellen in diesem Abschnitt werden die SQL- und XML-Begrenzungen beschrieben. Die Orientierung am restriktivsten Fall erleichtert den Entwurf von Anwendungsprogrammen, die sich problemlos portieren lassen.

In Tabelle 113 werden Begrenzungen in Byte aufgeführt. Diese Begrenzungen werden bei der Erstellung von Kennungen (oder Bezeichnern) nach einer Konvertierung aus der Anwendungscodepage in die Datenbankcodepage geprüft. Die Begrenzungen werden darüber hinaus auch beim Abrufen von Kennungen aus der Datenbank nach der Konvertierung aus der Datenbankcodepage in die Anwendungscode geprüft. Wenn während eines dieser Prozesse die Längenbegrenzung einer Kennung überschritten wird, wird die Kennung abgeschnitten oder es wird ein Fehler zurückgegeben.

Zeichenbegrenzungen sind je nach Codepage der Datenbank und der Codepage der Anwendung unterschiedlich. Da die Länge eines UTF-8-Zeichens zum Beispiel zwischen 1 und 4 Byte betragen kann, kann die Zeichenbegrenzung für eine Kennung in einer Unicode-Tabelle, deren Begrenzung bei 128 Byte liegt, je nach verwendeten Zeichen zwischen 32 und 128 Zeichen betragen. Wenn versucht wird, eine Kennung zu erstellen, die nach der Konvertierung in die Datenbankcodepage länger als die Begrenzung für die betreffende Tabelle ist, wird ein Fehler zurückgegeben.

Anwendungen, die Kennungen speichern, müssen in der Lage sein, die potenzielle Längenzunahme dieser Kennungen nach einer Codepagekonvertierung aufzufangen. Wenn Kennungen aus dem Katalog abgerufen werden, werden sie in die Codepage der Anwendung konvertiert. Die Konvertierung aus der Datenbankcodepage in die Anwendungscodepage kann dazu führen, dass eine Kennung länger wird und die Bytebegrenzung für die Tabelle überschreitet. Wenn eine von der Anwendung deklarierte Hostvariable nach einer Codepagekonvertierung nicht die gesamte Kennung speichern kann, wird die Kennung abgeschnitten. Wenn dies nicht akzeptabel ist, kann der Wert für die Größe der Hostvariablen erhöht werden, sodass die Hostvariable die gesamte Kennung aufnehmen kann.

Die gleichen Regeln gelten auch für DB2-Dienstprogramme, die Daten abrufen und in eine benutzerdefinierte Codepage konvertieren. Wenn ein DB2-Dienstprogramm, wie zum Beispiel EXPORT, Daten abruft und die Konvertierung in eine benutzerdefinierte Codepage (durch den EXPORT-Modifikator CODEPAGE oder durch die Registrierdatenbankvariable **DB2CODEPAGE**) erzwingt und die Daten die Begrenzung, die in der nachfolgenden Tabelle dokumentiert ist, infolge der Codepagekonvertierung überschreiten, wird möglicherweise ein Fehler zurückgegeben oder die Daten werden abgeschnitten.

Tabelle 113. Längenbegrenzungen von Kennungen

Beschreibung	Maximum in Byte
Aliasname	128
Attributname	128
Name einer Prüfrichtlinie	128
Berechtigungsname (nur Einzelbytezeichen sind zulässig)	128
Pufferpoolname	18

Tabelle 113. Längenbegrenzungen von Kennungen (Forts.)

Beschreibung	Maximum in Byte
Spaltenname[2]	128
Integritätsbedingungsname	128
Korrelationsname	128
Cursorname	128
Datenpartitionsname	128
Name einer Datenquellenspalte	255
Indexname einer Datenquelle	128
Datenquellenname	128
Tabellenname einer Datenquelle (*ferner-tabellenname*)	128
Name einer Datenbankpartitionsgruppe	128
Name einer Datenbankpartition	128
Name eines Ereignismonitors	128
Name eines externen Programms	128
Funktionszuordnungsname	128
Gruppenname	128
Hostkennung[1]	255
Kennung für Datenquellenbenutzer (*ferner-berechtigungsname*)	128
Kennung in SQL-Prozedur (Bedingungsname, For-Schleifen-Kennung, Kennsatz, Zeigerwert für Ergebnismenge, Anweisungsname und Variablenname)	128
Indexname	128
Indexerweiterungsname	18
Name einer Indexspezifikation	128
Kennsatzname	128
Uniform-Resource-Identifier (URI) eines Namensbereichs	1000
Kurzname	128
Paketname	128
Paketversions-ID	64
Parametername	128
Kennwort für Datenquellenzugriff	32
Prozedurname	128
Rollenname	128
Sicherungspunktname	128
Schemaname[2]	128
Name einer Sicherheitskennsatzkomponente	128
Name eines Sicherheitskennsatzes	128
Name einer Sicherheitsrichtlinie	128
Sequenzname	128
Servername (Aliasname einer Datenbank)	8
Bestimmter Name	128

Tabelle 113. Längenbegrenzungen von Kennungen (Forts.)

Beschreibung	Maximum in Byte
SQL-Bedingungsname	128
SQL-Variablenname	128
Anweisungsname	128
Speichergruppe	128
Tabellenname	128
Tabellenbereichsname	18
Umsetzungsgruppenname	18
Triggername	128
Name eines gesicherten Kontexts	128
Name einer Zuordnung von Inhaltstypen	18
Benutzerdefinierter Funktionsname	128
Benutzerdefinierter Methodenname	128
Name eines benutzerdefinierten Datentyps[2]	128
Sichtname	128
Wrappername	128
XML-Elementname, Attributname oder Präfixname	1000
Uniform-Resource-Identifier (URI) für Position des XML-Schemas	1000

Anmerkung:

1. Für einzelne Compiler für Hostprogrammiersprachen gilt möglicherweise eine noch restriktivere Begrenzung für Variablennamen.
2. Die Struktur des SQL-Deskriptorbereichs (SQLDA) ist auf das Speichern von 30 Byte langen Spaltennamen, 18 Byte langen Namen benutzerdefinierter Datentypen und 8 Byte langen Schemanamen für benutzerdefinierte Typen beschränkt. Da der SQL-Deskriptorbereich in der Anweisung DESCRIBE verwendet wird, müssen Anwendungen mit eingebettetem SQL, die die Anweisung DESCRIBE zum Abrufen von Informationen zu Spaltennamen oder Namen von benutzerdefinierten Datentypen verwenden, diese Begrenzungen einhalten.

Tabelle 114. Numerische Begrenzungen

Beschreibung	Begrenzung
Niedrigster SMALLINT-Wert	-32.768
Höchster SMALLINT-Wert	+32.767
Niedrigster INTEGER-Wert	-2.147.483.648
Höchster INTEGER-Wert	+2.147.483.647
Niedrigster BIGINT-Wert	-9.223.372.036.854.775.808
Höchster BIGINT-Wert	+9.223.372.036.854.775.807
Größte Dezimalgenauigkeit	31
Maximaler Exponent (E_{max}) für REAL-Werte	38
Niedrigster REAL-Wert	-3,402E+38
Höchster REAL-Wert	+3,402E+38

Tabelle 114. Numerische Begrenzungen (Forts.)

Beschreibung	Begrenzung
Minimaler Exponent (E_{min}) für REAL-Werte	-37
Niedrigster positiver REAL-Wert	+1,175E-37
Höchster negativer REAL-Wert	-1,175E-37
Maximaler Exponent (E_{max}) für DOUBLE-Werte	308
Niedrigster DOUBLE-Wert	-1,79769E+308
Höchster DOUBLE-Wert	+1,79769E+308
Minimaler Exponent (E_{min}) für DOUBLE-Werte	-307
Niedrigster positiver DOUBLE-Wert	+2,225E-307
Höchster negativer DOUBLE-Wert	-2,225E-307
Maximaler Exponent (E_{max}) für DECFLOAT(16)-Werte	384
Niedrigster DECFLOAT(16)-Wert[1]	-9,999999999999999E+384
Höchster DECFLOAT(16)-Wert	9,999999999999999E+384
Minimaler Exponent (E_{min}) für DECFLOAT(16)-Werte	-383
Niedrigster positiver DECFLOAT(16)-Wert	1,000000000000000E-383
Höchster negativer DECFLOAT(16)-Wert	-1,000000000000000E-383
Maximaler Exponent (E_{max}) für DECFLOAT(34)-Werte	6144
Niedrigster DECFLOAT(34)-Wert[1]	-9,999999999999999999999999999999999E+6144
Höchster DECFLOAT(34)-Wert	9,999999999999999999999999999999999E+6144
Minimaler Exponent (E_{min}) für DECFLOAT(34)-Werte	-6143
Niedrigster positiver DECFLOAT(34)-Wert	1,000000000000000000000000000000000E-6143
Höchster negativer DECFLOAT(34)-Wert	-1,000000000000000000000000000000000E-6143

Tabelle 114. Numerische Begrenzungen (Forts.)

Beschreibung	Begrenzung
Anmerkung: 1. Dies sind die Begrenzungen für normale Gleitkommadezimalzahlen. Gültige Gleitkommadezimalwerte sind außerdem die Sonderwerte NAN, -NAN, SNAN, -SNAN, INFINITY und -INFINITY. Darüber hinaus gehören zu den gültigen Werten subnormale Zahlen. Subnormale Zahlen sind Zahlen ungleich null, deren angepasste Exponenten kleiner als E_{min} sind. Bei einer subnormalen Zahl ist der Minimalwert des Exponenten gleich E_{min} - (*genauigkeit*-1), als E_{tiny} bezeichnet, wobei *genauigkeit* die Genauigkeit ist, mit der gearbeitet wird (16 oder 34). Das heißt, subnormale Zahlen erweitern den Bereich der Zahlen nahe null um 15 oder 33 Größenordnungen für DECFLOAT(16) bzw. DECFLOAT(34). Subnormale Zahlen unterscheiden sich von normalen Zahlen, weil die maximale Anzahl von Stellen kleiner ist als die Genauigkeit, mit der gearbeitet wird (16 bzw. 34). Gleitkommadezimalzahlen können subnormale Zahlen nicht mit derselben Präzision wie normale Zahlen darstellen. Die niedrigste positive subnormale Zahl für DECFLOAT(34) ist $1x10^{-6176}$, die nur eine Stelle hat, während für DECFLOAT(34) die niedrigste positive normale Zahl $1{,}000000000000000000000000000000000x10^{-6143}$ ist, die 34 Stellen enthält. Die niedrigste positive subnormale Zahl für DECFLOAT(16) ist $1x10^{-398}$.	

Tabelle 115. Zeichenfolgebegrenzungen

Beschreibung	Begrenzung
Maximale Länge von CHAR (in Byte)	254
Maximale Länge von VARCHAR (in Byte)	32 672
Maximale Länge von LONG VARCHAR (in Byte)[1]	32 700
Maximale Länge von CLOB (in Byte)	2 147 483 647
Maximale Länge von serialisierter XML (in Byte)	2 147 483 647
Maximale Länge von GRAPHIC (in Doppelbytezeichen)	127
Maximale Länge von VARGRAPHIC (in Doppelbytezeichen)	16 336
Maximale Länge von LONG VARGRAPHIC (in Doppelbytezeichen)[1]	16 350
Maximale Länge von DBCLOB (in Doppelbytezeichen)	1 073 741 823
Maximale Länge von BLOB (in Byte)	2 147 483 647
Maximale Länge der Zeichenkonstanten	32 672
Maximale Länge der Grafikkonstanten	16 336
Maximale Länge der verknüpften Zeichenfolge	2 147 483 647
Maximale Länge der verknüpften Grafikzeichenfolge	1 073 741 823
Maximale Länge der verknüpften Binärzeichenfolge	2 147 483 647
Maximale Anzahl von Hexadezimalkonstantenziffern	32 672
Größte Instanz eines Spaltenobjekts mit strukturiertem Typ während der Laufzeit (in Gigabyte)	1
Maximale Größe eines Katalogkommentars (in Byte)	254
Anmerkung: 1. Die Datentypen LONG VARCHAR und LONG VARGRAPHIC sind veraltet und werden in einem zukünftigen Release möglicherweise entfernt.	

Tabelle 116. XML-Begrenzungen

Beschreibung	Begrenzung
Maximale Verschachtelungstiefe eines XML-Dokuments (in Stufen)	125
Maximale Größe eines XML-Schemadokuments (in Byte)	31 457 280

Tabelle 117. Begrenzungen für Datums- und Uhrzeitangaben

Beschreibung	Begrenzung
Niedrigster DATE-Wert	0001-01-01
Höchster DATE-Wert	9999-12-31
Niedrigster TIME-Wert	00:00:00
Höchster TIME-Wert	24:00:00
Niedrigster TIMESTAMP-Wert	0001-01-01-00.00.00.000000000000
Höchster TIMESTAMP-Wert	9999-12-31-24.00.00.000000000000
Niedrigste TIMESTAMP-Genauigkeit	0
Höchste TIMESTAMP-Genauigkeit	12

Tabelle 118. Begrenzungen des Datenbankmanagers

Kategorie	Beschreibung	Begrenzung
Anwendungen	Maximale Anzahl von Deklarationen von Hostvariablen in einem vorkompilierten Programm[3]	Speicher
	Maximale Länge eines Hostvariablenwerts (in Byte)	2 147 483 647
	Maximale Anzahl deklarierter Cursor in einem Programm	Speicher
	Maximale Anzahl von Zeilen, die in einer Arbeitseinheit geändert wurden	Speicher
	Maximale Anzahl von Cursorn, die zu einem bestimmten Zeitpunkt geöffnet wurden	Speicher
	Maximale Anzahl von Verbindungen pro Prozess in einem DB2-Client	512
	Maximale Anzahl von gleichzeitig geöffneten LOB-Querverweisen in einer Transaktion	4 194 304
	Maximale Größe eines SQL-Deskriptorbereichs (in Byte)	Speicher
	Maximale Anzahl vorbereiteter Anweisungen	Speicher
Pufferpools	Maximale Anzahl NPAGES in einem Pufferpool für 32-Bit-Releases	1 048 576
	Maximale Anzahl NPAGES in einem Pufferpool für 64-Bit-Releases	2 147 483 647
	Maximale Gesamtgröße aller Pufferpoolslots (4 KB)	2 147 483 646

Tabelle 118. Begrenzungen des Datenbankmanagers (Forts.)

Kategorie	Beschreibung	Begrenzung
Gemeinsamer Zugriff	Maximale Anzahl gleichzeitig angemeldeter Benutzer eines Servers[4]	64 000
	Maximale Anzahl gleichzeitig angemeldeter Benutzer pro Instanz	64 000
	Maximale Anzahl gleichzeitig ausgeführter Anwendungen pro Datenbank	60 000
	Maximale Anzahl von Datenbanken pro Instanz, die gleichzeitig benutzt werden können	256
Integritätsbedingungen	Maximale Anzahl von Integritätsbedingungen für eine Tabelle	Speicher
	Maximale Anzahl von Spalten in einer UNIQUE-Integritätsbedingung (unterstützt über einen UNIQUE-Index)	64
	Maximale kombinierte Länge der Spalten in einer UNIQUE-Integritätsbedingung (unterstützt über einen UNIQUE-Index - in Byte)[8]	8192
	Maximale Anzahl von verweisenden Spalten in einem Fremdschlüssel	64
	Maximale kombinierte Länge von verweisenden Spalten in einem Fremdschlüssel (in Byte)[8]	8192
	Maximale Länge einer Prüfbedingungsangabe (in Byte)	65 535
Datenbanken	Maximale Anzahl von Datenbankpartitionen	999
	Maximale Anzahl Member in einer DB2 pureScale-Umgebung	128

Tabelle 118. Begrenzungen des Datenbankmanagers (Forts.)

Kategorie	Beschreibung	Begrenzung
Indizes	Maximale Anzahl von Indizes für eine Tabelle	32 767 oder Speicher
	Maximale Anzahl von Spalten in einem Indexschlüssel	64
	Maximale Länge eines Indexschlüssels einschließlich des gesamten Systemaufwands[6] [8]	*indexseitengröße*/4
	Maximale Länge eines variablen Indexschlüsselteils (in Byte)[7]	1022 oder Speicher
	Maximale Größe eines Indexes pro Datenbankpartition in einem SMS-Tabellenbereich (in Terabyte)[6]	64
	Maximale Größe eines Indexes pro Datenbankpartition in einem regulären DMS-Tabellenbereich (in Gigabyte)[6]	512
	Maximale Größe eines Indexes pro Datenbankpartition in einem großen DMS-Tabellenbereich (in Terabyte)[6]	64
	Maximale Länge eines variablen Indexschlüsselteils für einen Index für XML-Daten (in Byte)[7]	*seitengröße*/4 - 207
Protokollsätze	Maximale Protokollfolgenummer (LSN)	0xFFFF FFFF FFFF FFFF
Überwachung	Maximale Anzahl gleichzeitig aktiver Ereignismonitore	128
	In einer DB2-Umgebung mit partitionierten Datenbanken: maximale Anzahl gleichzeitig aktiver globaler Ereignismonitore	32
Routinen	Maximale Anzahl von Parametern in einer Prozedur mit LANGUAGE SQL	32 767
	Maximale Anzahl von Parametern in einer externen Prozedur mit PROGRAM TYPE MAIN	32 767
	Maximale Anzahl von Parametern in einer externen Prozedur mit PROGRAM TYPE SUB	90
	Maximale Anzahl von Parametern in einem Cursorwertkonstruktor	32 767
	Maximale Anzahl von Parametern in einer benutzerdefinierten Funktion	90
	Maximale Anzahl von Verschachtelungsebenen für Routinen	64
	Maximale Anzahl von Schemata im SQL-Pfad	64
	Maximale Länge des SQL-Pfads (in Byte)	2048

Tabelle 118. Begrenzungen des Datenbankmanagers (Forts.)

Kategorie	Beschreibung	Begrenzung
Sicherheit	Maximale Anzahl von Elementen in einer Sicherheitskennsatzkomponente des Typs 'set' oder 'tree'	64
	Maximale Anzahl von Elementen in einer Sicherheitskennsatzkomponente des Typs 'array'	65 535
	Maximale Anzahl von Sicherheitskennsatzkomponenten in einer Sicherheitsrichtlinie	16
SQL	Maximale Gesamtlänge einer SQL-Anweisung (in Byte)	2 097 152
	Maximale Anzahl von Tabellen, auf die in einer SQL-Anweisung oder einer Sicht verwiesen wird	Speicher
	Maximale Anzahl von Verweisen auf Hostvariablen in einer SQL-Anweisung	32 767
	Maximale Anzahl von Konstanten in einer Anweisung	Speicher
	Maximale Anzahl von Elementen in einer SELECT-Liste[6]	1012
	Maximale Anzahl von Vergleichselementen in einer Klausel WHERE oder HAVING	Speicher
	Maximale Anzahl von Spalten in einer Klausel GROUP BY[6]	1012
	Maximale Gesamtlänge der Spalten in einer Klausel GROUP BY (in Byte)[6]	32 677
	Maximale Anzahl von Spalten in einer Klausel ORDER BY[6]	1012
	Maximale Gesamtlänge der Spalten in einer Klausel ORDER BY (in Byte)[6]	32 677
	Höchste Ebene der Unterabfragenverschachtelung	Speicher
	Maximale Anzahl von Unterabfragen in einer einzelnen Anweisung	Speicher
	Maximale Anzahl von Werten in einer Einfügeoperation[6]	1012
Speicher- gruppen	Maximale Anzahl von Speichergruppen in einer Datenbank	256
	Maximale Anzahl von Speicherpfaden in einer Speichergruppe	128
	Maximale Länge des Speicherpfads (in Byte)	175

Tabelle 118. Begrenzungen des Datenbankmanagers (Forts.)

Kategorie	Beschreibung	Begrenzung
Tabellen und Sichten	Maximale Anzahl von Spalten in einer Tabelle[6]	1012
	Maximale Anzahl von Spalten in einer Sicht[1]	5000
	Maximale Anzahl von Spalten in einer Datenquellentabelle oder einer Sicht, auf die mit einem Kurznamen verwiesen wird	5000
	Maximale Anzahl von Spalten in einem Verteilungsschlüssel[5]	500
	Maximale Länge einer Zeile einschließlich des gesamten Systemaufwands[2 6]	32 677
	Maximale Anzahl von Zeilen in einer nicht partitionierten Tabelle (pro Datenbankpartition)	128×10^{10}
	Maximale Anzahl von Zeilen in einer Datenpartition (pro Datenbankpartition)	128×10^{10}
	Maximale Größe einer Tabelle pro Datenbankpartition in einem regulären Tabellenbereich (in Gigabyte)[3 6]	512
	Maximale Größe einer Tabelle pro Datenbankpartition in einem großen DMS-Tabellenbereich (in Terabyte)[6]	64
	Maximale Anzahl von Datenpartitionen für eine einzelne Tabelle	32 767
	Maximale Anzahl von Tabellenpartitionierungsspalten	16
	Maximale Anzahl von Feldern in einem benutzerdefinierten Zeilentyp	1012
Tabellen- bereiche	Maximale Größe eines LOB-Objekts pro Tabelle oder pro Tabellenpartition (in Terabyte)	4
	Maximale Größe eines LF-Objekts pro Tabelle oder pro Tabellenpartition (in Terabyte)	2
	Maximale Anzahl von Tabellenbereichen in einer Datenbank	32 768
	Maximale Anzahl von Tabellen in einem SMS-Tabellenbereich	65 532
	Maximale Größe eines regulären DMS-Tabellenbereichs (in Gigabyte) [3 6]	512
	Maximale Größe eines großen DMS-Tabellenbereichs (in Terabyte) [3 6]	64
	Maximale Größe eines DMS-Tabellenbereichs für temporäre Tabellen (in Terabyte) [3 6]	64
	Maximale Anzahl von Tabellenobjekten in einem DMS-Tabellenbereich	Weitere Informationen hierzu finden Sie in Tabelle 119 auf Seite 615.

Tabelle 118. Begrenzungen des Datenbankmanagers (Forts.)

Kategorie	Beschreibung	Begrenzung
Trigger	Maximale Laufzeitverschachtelungstiefe von hintereinandergeschalteten Triggern	16
Benutzerdefinierte Typen	Maximale Anzahl von Attributen in einem strukturierten Typ	4082
Workload Manager	Maximale Anzahl von Servicesuperklassen pro Datenbank	64
	Maximale Anzahl von benutzerdefinierten Serviceunterklassen pro Servicesuperklasse	61

Anmerkung:

1. Dieser Maximalwert kann erreicht werden, wenn in der Anweisung CREATE VIEW ein Join verwendet wird. Die Auswahl aus einer solchen Sicht unterliegt den Begrenzungen der meisten Elemente in einer SELECT-Liste.
2. Die eigentlichen Daten für BLOB-, CLOB-, LONG VARCHAR-, DBCLOB- und LONG VARGRAPHIC-Spalten sind bei dieser Zählung nicht berücksichtigt. Allerdings belegen die Informationen zur Position dieser Daten innerhalb der Zeile eine gewisse Menge an Speicherplatz.
3. Die angezeigten Nummern stellen durch die Architektur bedingte Begrenzungen und Approximationen dar. Die in der Praxis geltenden Begrenzungen können darunter liegen.
4. Der tatsächliche Wert wird durch die Konfigurationsparameter **max_connections** und **max_coordagents** des Datenbankmanagers gesteuert.
5. Dies ist eine durch die Architektur bedingte Begrenzung. Als praktische Begrenzung sollte die Begrenzung benutzt werden, die für die meisten Spalten in einem Indexschlüssel gilt.
6. Weitere Informationen zu speziellen Werten für bestimmte Seitengrößen finden Sie in Tabelle 119.
7. Eine Begrenzung erfolgt lediglich durch den längsten Indexschlüssel einschließlich des gesamten Systemaufwands (in Byte). Wenn die Anzahl der Indexschlüsselteile ansteigt, reduziert sich die maximale Länge der einzelnen Schlüsselteile.
8. Das Maximum kann abhängig von den Indexoptionen geringer sein.

Tabelle 119. Seitengrößenspezifische Begrenzungen des Datenbankmanagers

Beschreibung	Seitengrößenbegrenzung - 4 KB	Seitengrößenbegrenzung - 8 KB	Seitengrößenbegrenzung - 16 KB	Seitengrößenbegrenzung - 32 KB
Maximale Anzahl von Tabellenobjekten in einem DMS-Tabellenbereich[1]	51 971[2] 53 212[3]	53 299	53 747	54 264
Maximale Anzahl von Spalten in einer Tabelle	500	1012	1012	1012
Maximale Länge einer Zeile einschließlich des gesamten Systemaufwands	4005	8101	16 293	32 677

Tabelle 119. Seitengrößenspezifische Begrenzungen des Datenbankmanagers (Forts.)

Beschreibung	Seitengrößenbegrenzung - 4 KB	Seitengrößenbegrenzung - 8 KB	Seitengrößenbegrenzung - 16 KB	Seitengrößenbegrenzung - 32 KB
Maximale Größe einer Tabelle pro Datenbankpartition in einem regulären Tabellenbereich (in Gigabyte)	64	128	256	512
Maximale Größe einer Tabelle pro Datenbankpartition in einem großen DMS-Tabellenbereich (in Terabyte)	8	16	32	64
Maximale Länge eines Indexschlüssels einschließlich des gesamten Systemaufwands (in Byte)	1024	2048	4096	8192
Maximale Größe eines Indexes pro Datenbankpartition in einem SMS-Tabellenbereich (in Terabyte)	8	16	32	64
Maximale Größe eines Indexes pro Datenbankpartition in einem regulären DMS-Tabellenbereich (in Gigabyte)	64	128	256	512
Maximale Größe eines Indexes pro Datenbankpartition in einem großen DMS-Tabellenbereich (in Terabyte)	8	16	32	64
Maximale Größe eines regulären DMS-Tabellenbereichs (in Gigabyte)	64	128	256	512
Maximale Größe eines großen DMS-Tabellenbereichs (in Terabyte)	8	16	32	64
Maximale Größe eines DMS-Tabellenbereichs für temporäre Tabellen (in Terabyte)	8	16	32	64
Maximale Anzahl von Elementen in einer SELECT-Liste	500[4]	1012	1012	1012
Maximale Anzahl von Spalten in einer Klausel GROUP BY	500	1012	1012	1012
Maximale Gesamtlänge der Spalten in einer Klausel GROUP BY (in Byte)	4005	8101	16 293	32 677

Tabelle 119. Seitengrößenspezifische Begrenzungen des Datenbankmanagers (Forts.)

Beschreibung	Seitengrößenbegrenzung - 4 KB	Seitengrößenbegrenzung - 8 KB	Seitengrößenbegrenzung - 16 KB	Seitengrößenbegrenzung - 32 KB
Maximale Anzahl von Spalten in einer Klausel ORDER BY	500	1012	1012	1012
Maximale Gesamtlänge der Spalten in einer Klausel ORDER BY (in Byte)	4005	8101	16 293	32 677
Maximale Anzahl von Werten in einer Einfügeoperation	500	1012	1012	1012
Maximale Anzahl von SET-Klauseln in einer einzigen Aktualisierungsoperation	500	1012	1012	1012
Maximale Anzahl von Datensätzen pro Seite für einen regulären Tabellenbereich (REGULAR)	251	253	254	253
Maximale Anzahl von Datensätzen pro Seite für einen großen Tabellenbereich (LARGE)	287	580	1165	2335

Anmerkung:

1. Tabellenobjekte umfassen Tabellendaten, Indizes, LONG VARCHAR- oder LONG VARGRAPHIC-Spalten sowie LOB-Spalten. Tabellenobjekte, die sich im selben Tabellenbereich befinden wie die Tabellendaten, werden bei der Ermittlung der Begrenzung nicht gesondert berücksichtigt. Jedes Tabellenobjekt, das sich in einem anderen Tabellenbereich als die Tabellendaten befindet, trägt jedoch zum Erreichen der Begrenzung für die einzelnen Tabellenobjekttypen pro Tabelle in dem Tabellenbereich bei, in dem sich das Tabellenobjekt befindet.
2. Bei einer Speicherbereichsgröße von 2 Seiten.
3. Bei einer Speicherbereichsgröße, die nicht 2 Seiten beträgt.
4. In Fällen, in denen der einzige Tabellenbereich für temporäre Systemtabellen eine Größe von 4 KB aufweist und ein Datenüberlauf in den Sortierpuffer stattfindet, wird ein Fehler generiert. Wenn ausreichend Hauptspeicherplatz für die Ergebnismenge vorhanden ist, tritt kein Fehler auf.

Kapitel 22. Registrierdatenbank- und Umgebungsvariablen

Umgebungsvariablen und Profilregistrierdatenbanken

DB2-Datenbankumgebungen werden mithilfe von Umgebungsvariablen und Registrierdatenbankvariablen gesteuert. Mit den DB2-Profilregistrierdatenbanken können Sie Informationen zu Variablen und Instanzen aktualisieren.

Vor Einführung der DB2-Datenbankprofilregistrierdatenbanken waren für das Definieren von Umgebungsvariablen die Angabe eines Werts für die jeweilige Umgebungsvariable und das Neustarten des Computers erforderlich. Jetzt können Sie die meisten Variablen, die sich auf die DB2-Datenbankumgebung auswirken, mit den DB2-Profilregistrierdatenbanken steuern.

Verwenden Sie die Profilregistrierdatenbanken zum Steuern der Umgebungsvariablen von einem einzigen Computer aus. Über die verschiedenen Profile stehen unterschiedliche Unterstützungsebenen bereit. Die Umgebungsvariablen können mit dem DB2-Verwaltungsserver über Fernzugriff verwaltet werden.

Eine DB2-Datenbank wird durch folgende Profilregistrierdatenbanken gesteuert:

- Die DB2-Profilregistrierdatenbank auf Instanzebene enthält Registrierdatenbankvariablen für eine Instanz. Die in dieser Registrierdatenbank definierten Werte überschreiben die entsprechenden Einstellungen in der globalen Registrierdatenbank.
- Die DB2-Profilregistrierdatenbank auf globaler Ebene enthält Einstellungen, die verwendet werden, wenn eine Registrierdatenbankvariable nicht für eine Instanz definiert wird. Alle Instanzen für eine bestimmte Kopie von DB2 Enterprise Server Edition können auf diese Registrierdatenbank zugreifen.
- Die DB2-Profilregistrierdatenbank auf Instanzknotenebene enthält Variableneinstellungen, die für eine Datenbankpartition in einer Umgebung mit partitionierten Datenbanken spezifisch sind. Die in dieser Registrierdatenbank definierten Werte überschreiben die entsprechenden Einstellungen auf der Instanzebene und der globalen Ebene.
- Die DB2-Profilregistrierdatenbank auf Benutzerebene enthält Einstellungen, die für jeden Benutzer spezifisch sind. Die in dieser Registrierdatenbank definierten Werte überschreiben die entsprechenden Einstellungen in den anderen Registrierdatenbanken.

Das DB2-Datenbanksystem konfiguriert die Betriebsumgebung, wobei die Registrierungswerte und die Umgebungsvariablen geprüft und in folgender Reihenfolge auflöst werden:

1. Umgebungsvariablen, die außerhalb der Profilregistrierdatenbanken definiert werden.
2. Registrierdatenbankvariablen, die mit dem Profil auf Benutzerebene definiert werden.
3. Registrierdatenbankvariablen, die mit dem Profil auf Instanzknotenebene definiert werden.
4. Registrierdatenbankvariablen, die mit dem Profil auf Instanzebene definiert werden.
5. Registrierdatenbankvariablen, die mit dem Profil auf globaler Ebene definiert werden.

Die DB2-Instanzprofilregistrierdatenbank enthält eine Liste aller Instanzen, die der aktuellen Kopie zugeordnet sind. Für jede DB2-Kopie ist eine Liste vorhanden. Sie können die vollständige Liste aller auf dem System verfügbaren Instanzen durch Ausführen des Befehls **db2ilist** anzeigen. Diese Profilregistrierdatenbank enthält keine Variableneinstellungen.

Speicherpositionen und Berechtigungsvoraussetzungen für Profilregistrierdatenbanken

Die DB2-Profilregistrierdatenbanken weisen auf jedem Betriebssystem unterschiedliche Speicherpositionen und Berechtigungsvoraussetzungen auf. Eine Berechtigung ist für die Aktualisierung der Werte von Variablen in jeder Profilregistrierdatenbank erforderlich.

Tabelle 120. Speicherpositionen und Berechtigungsvoraussetzungen für Profilregistrierdatenbanken

Profilregistrierdatenbank	Speicherposition unter Windows	Speicherposition unter Linux und UNIX	Berechtigungsvoraussetzungen unter Linux und UNIX	Berechtigungsvoraussetzungen unter Windows
Profilregistrierdatenbank der Instanzebene	\HKEY_LOCAL_computer\SOFTWARE\IBM\DB2\PROFILES*instanzname*	*instanzausgangsverzeichnis*/sqllib/profile.env Dabei ist *instanzausgangsverzeichnis* der Ausgangspfad des Instanzeigners.	-rw-rw-r-- *instanzeigner instanzeignergruppe* profile.env	Sie müssen Mitglied der DB2-Administratorgruppe (DB2ADMNS) sein.
Profilregistrierdatenbank auf globaler Ebene	\HKEY_LOCAL_computer\SOFTWARE\IBM\DB2\GLOBAL_PROFILE	Bei Installationen mit Rootberechtigung: /var/db2/global.reg Bei Installationen ohne Rootberechtigung: *ausgangsverzeichnis*/sqllib/global.reg	Zum Ändern einer globalen Registrierdatenbankvariablen in Installationen mit Rootberechtigung müssen Sie mit Rootberechtigung angemeldet sein.	Sie müssen Mitglied der DB2-Administratorgruppe (DB2ADMNS) sein.
Profilregistrierdatenbank auf Instanzknotenebene	...\SOFTWARE\IBM\DB2\PROFILES*instanzname*\NODES*knotennummer*	*instanzausgangsverzeichnis*/sqllib/nodes/*knotennummer*.env Dabei ist *instanzausgangsverzeichnis* der Ausgangspfad des Instanzeigners.	Für das Verzeichnis mit der Datei: drwxrwsr-w *instanzeigner instanzeignergruppe* Knoten Für die Datei: -rw-rw-r-- *instanzeigner instanzeignergruppe knotennummer*.env	Sie müssen Mitglied der DB2-Administratorgruppe (DB2ADMNS) sein.
Profilregistrierdatenbank auf Benutzerebene	LDAP-Verzeichnis	Nicht zutreffend	Nicht zutreffend	Sie müssen Mitglied der DB2-Administratorgruppe (DB2ADMNS) sein.
Instanzprofilregistrierdatenbank	\HKEY_LOCAL_computer\SOFTWARE\IBM\DB2\PROFILES*instanzname*	Bei Installationen mit Rootberechtigung: /var/db2/global.reg Bei Installationen ohne Rootberechtigung: *ausgangsverzeichnis*/sqllib/global.reg	Keine erforderlich	Keine erforderlich

Definieren von Registrierdatenbank- und Umgebungsvariablen

Die meisten Umgebungsvariablen werden mit dem Befehl **db2set** in den DB2-Datenbankprofilregistrierdatenbanken definiert. Für die wenigen Variablen, die außerhalb der Profilregistrierdatenbanken definiert werden, sind je nach dem verwendeten Betriebssystem unterschiedliche Befehle erforderlich.

Vorbereitende Schritte

Stellen Sie sicher, dass Sie über die Berechtigungen verfügen, die für die Definition von Registrierdatenbankvariablen erforderlich sind.

Unter Linux- und UNIX-Betriebssystemen müssen Sie über folgende Berechtigungen verfügen:

- Berechtigung SYSADM zum Definieren von Variablen in der Registrierdatenbank auf Instanzebene
- Rootberechtigung zum Definieren von Variablen in der Registrierdatenbank auf globaler Ebene

Unter Windows-Betriebssystemen müssen Sie über eine der folgenden Berechtigungen verfügen:

- Berechtigung eines lokalen Administrators
- Berechtigung SYSADM mit folgenden Bedingungen:
  - Wenn die erweiterte Sicherheit aktiviert ist, müssen SYSADM-Benutzer zur Gruppe DB2ADMNS gehören.
  - Wenn die erweiterte Sicherheit nicht aktiviert ist, können SYSADM-Benutzer Aktualisierungen vornehmen, wenn ihnen die entsprechenden Berechtigungen in der Windows-Registrierdatenbank erteilt wurden.

Informationen zu diesem Vorgang

Wenn Sie mit dem Befehl **db2set** Variablen in den Profilregistrierdatenbanken definieren, müssen Sie den Computer nicht erneut starten, damit die Variablenwerte wirksam werden. Änderungen wirken sich jedoch nicht auf aktuell aktive DB2-Anwendungen oder auf aktive Benutzer aus. Die DB2-Registrierdatenbank wendet die aktualisierten Informationen auf diejenigen DB2-Serverinstanzen und DB2-Anwendungen an, die nach der Durchführung der Änderungen gestartet werden.

Wenn DB2-Variablen außerhalb der Registrierdatenbank definiert werden, können sie nicht über Fernzugriff verwaltet werden. Darüber hinaus müssen Sie den Computer erneut starten, damit die Variablenwerte wirksam werden.

Die DB2-Umgebungsvariablen **DB2INSTANCE** und **DB2NODE** werden nicht in den DB2-Profilregistrierdatenbanken gespeichert. Informationen zum Definieren dieser Variablen können Sie den Abschnitten zum Definieren von Umgebungsvariablen außerhalb der Profilregistrierdatenbanken entnehmen.

Unter Linux- und UNIX-Betriebssystemen wird die Profilregistrierdatenbank auf Instanzebene in der Textdatei `profile.env` gespeichert. Wenn zwei oder mehr Benutzer eine Registrierdatenbankvariable nahezu gleichzeitig mit dem Befehl **db2set** definieren, wird die Größe dieser Datei auf null reduziert. Darüber hinaus zeigt die Ausgabe des Befehls **db2set -all** inkonsistente Werte.

Vorgehensweise

Gehen Sie wie folgt vor, um eine Registrierdatenbankvariable zu definieren:

Setzen Sie den Befehl **db2set** mit den relevanten Parametern ab.
Die folgende Tabelle enthält einige der Methoden, wie Sie Registrierdatenbankvariablen mit dem Befehl **db2set** definieren können. Weitere Informationen zu den Parametern und der Syntax dieses Befehls finden Sie im Referenzabschnitt für den Befehl **db2set**.

Tabelle 121. Häufige Befehle für die Definition von Registrierdatenbankvariablen

Gewünschte Aktion	Befehl
Definieren einer Registrierdatenbankvariablen für die aktuelle oder Standardinstanz	`db2set name_der_registrierdatenbankvariablen=neuer_wert`
Definieren einer Registrierdatenbankvariablen für alle Datenbanken einer Instanz	`db2set name_der_registrierdatenbankvariablen=neuer_wert -i instanzname`
Definieren einer Registrierdatenbankvariablen für eine bestimmte Datenbankpartition einer Instanz	`db2set name_der_registrierdatenbankvariablen=neuer_wert -i instanzname datenbankpartitionsnummer`
Definieren einer Registrierdatenbankvariablen für alle Instanzen einer DB2 Enterprise Server Edition-Installation	`db2set name_der_registrierdatenbankvariablen=neuer_wert -g`
Definieren einer Registrierdatenbankvariablen auf Benutzerebene in einer LDAP-Umgebung	`db2set name_der_registrierdatenbankvariablen=neuer_wert -ul`
Definieren einer Registrierdatenbankvariablen auf globaler Ebene in einer LDAP-Umgebung **DB2LDAP_KEEP_CONNECTION** und **DB2LDAP_SEARCH_SCOPE** sind die einzigen beiden Registrierdatenbankvariablen, die auf globaler LDAP-Ebene definiert werden können.	`db2set name_der_registrierdatenbankvariablen=neuer_wert -gl`

Tipp: Wenn eine Registrierdatenbankvariable boolesche Werte als Argumente erfordert, sind einerseits die Werte `YES`, `1`, `TRUE` und `ON` sowie andererseits die Werte `NO`, `0`, `FALSE` und `OFF` äquivalent. Für jede Variable können Sie einen beliebigen der entsprechenden äquivalenten Werte angeben.

Definieren von Umgebungsvariablen außerhalb der Profilregistrierdatenbanken unter Windows

Unter Windows-Betriebssystemen können die Umgebungsvariablen **DB2INSTANCE**, **DB2NODE** und **DB2PATH** nur außerhalb der Profilregistrierdatenbanken definiert werden. Sie müssen lediglich die Variable **DB2PATH** definieren.

Informationen zu diesem Vorgang

Unter Windows-Betriebssystemen werden die folgenden Umgebungsvariablen außerhalb der Profilregistrierdatenbanken definiert:

- Die Umgebungsvariable **DB2INSTANCE** gibt die Instanz an, die standardmäßig aktiv ist. Wenn diese Variable nicht definiert ist, verwendet der DB2-Datenbankmanager den Wert der Variablen **DB2INSTDEF** als aktuelle Instanz.
- Die Umgebungsvariable **DB2NODE** gibt den logischen Zielknoten eines Datenbankpartitionsservers an, an den Anforderungen weitergeleitet werden.
- Die Umgebungsvariable **DB2PATH** gibt das Verzeichnis an, in dem das DB2-Datenbankprodukt auf 32-Bit-Windows-Betriebssystemen installiert ist.

Wenn Sie andere Variablen definieren wollen, müssen diese in mindestens einer Profilregistrierdatenbank definiert werden.

Der Wert einer Umgebungsvariablen kann mit dem Befehl **echo** ermittelt werden. Wenn Sie beispielsweise den Wert der Umgebungsvariablen **DB2NODE** prüfen wollen, setzen Sie folgenden Befehl ab:

```
echo %db2path%
```

Vorgehensweise

Gehen Sie wie folgt vor, um eine Umgebungsvariable außerhalb der Profilregistrierdatenbanken zu definieren:

Definieren Sie eine Umgebungsvariable mit einer der folgenden Optionen.

Option	Bezeichnung
Definieren Sie die Umgebungsvariable auf Instanzebene.	1. Führen Sie die passende Vorgehensweise für Ihr Windows-Betriebssystem aus. 2. Starten Sie den Computer erneut.
Definieren Sie die Umgebungsvariable für die aktuelle Sitzung.	Setzen Sie den folgenden Befehl ab: `set` *`name_der_umgebungsvariablen=neuer_wert`* `db2start`
Definieren Sie die Umgebungsvariable für die aktuelle Sitzung für eine C-Shell.	Setzen Sie den folgenden Befehl ab: `setenv` *`name_der_umgebungsvariablen neuer_wert`*

Definieren von Umgebungsvariablen außerhalb der Profilregistrierdatenbanken unter Linux- und UNIX-Betriebssystemen

Unter Linux- und UNIX-Betriebssystemen müssen Sie die Systemumgebungsvariable **DB2INSTANCE** außerhalb der Profilregistrierdatenbanken definieren. Wenn Sie andere Variablen definieren wollen, müssen diese in mindestens einer Profilregistrierdatenbank definiert werden.

Informationen zu diesem Vorgang

Mit den Scripts `db2profile` (für Bourne- oder Korn-Shell) und `db2cshrc` (für C-Shell) können Sie die Variable **DB2INSTANCE** auf einen von Ihnen angegebenen Instanznamen setzen. Diese Scripts sind im Verzeichnis *`instanzausgangsverzeichnis`*`/sqllib` enthalten. Dabei ist *instanzausgangsverzeichnis* das Ausgangsverzeichnis des Instanzeigners.

Instanzeigner und Benutzer mit der Berechtigung SYSADM können diese Scripts für alle Benutzer einer Instanz anpassen. Alternativ können Benutzer ein Script kopieren und anpassen und anschließend das Script direkt aufrufen oder ihrer Datei `.profile` oder `.login` hinzufügen.

Definieren Sie Variablen, die vom DB2-Datenbankmanager nicht unterstützt werden, in den Scriptdateien `userprofile` und `usercshrc`. Diese Dateien sind auch im Verzeichnis *`instanzausgangsverzeichnis`*`/sqllib` enthalten.

Vorgehensweise

Gehen Sie wie folgt vor, um eine Umgebungsvariable außerhalb der Profilregistrierdatenbanken zu definieren:

Definieren Sie eine Umgebungsvariable mit einer der folgenden Methoden.

Option	Bezeichnung
Definieren Sie die Umgebungsvariable auf Instanzebene für eine Bourne- oder Korn-Shell.	Führen Sie das Script `db2profile` aus.
Definieren Sie die Umgebungsvariable auf Instanzebene für eine C-Shell.	Führen Sie das Script `db2cshrc` aus.
Definieren Sie die Umgebungsvariable für die aktuelle Sitzung für eine Bourne-Shell.	Setzen Sie den folgenden Befehl ab: `export name_der_umgebungsvariablen=neuer_wert`
Definieren Sie die Umgebungsvariable für die aktuelle Sitzung für eine C-Shell.	Setzen Sie den folgenden Befehl ab: `setenv name_der_umgebungsvariablen neuer_wert`

Definieren Sie die Umgebungsvariable für die aktuelle Sitzung für eine Korn-Shell.	Setzen Sie den folgenden Befehl ab:
	`name_der_umgebungsvariablen=neuer_wert`
	`export name_der_umgebungsvariablen`

Ermitteln der aktuellen Instanz

Die meisten der von Ihnen abgesetzten Befehle oder ausgeführten Konfigurationsänderungen gelten standardmäßig für die aktuelle Instanz. Diese können Sie ermitteln, indem Sie die Werte bestimmter Umgebungsvariablen prüfen.

Informationen zu diesem Vorgang

Wenn Sie Befehle zum Starten oder Stoppen des Datenbankmanagers einer Instanz ausführen, wendet der Datenbankmanager den jeweiligen Befehl auf die aktuelle Instanz an. Zum Ermitteln der aktuellen Instanz prüft der Datenbankmanager die Werte bestimmter Umgebungsvariablen in der folgenden Reihenfolge:

1. Den Wert der Umgebungsvariablen **DB2INSTANCE** für die aktuelle Sitzung.
2. Den Wert der Systemumgebungsvariablen **DB2INSTANCE**.
3. Unter Windows-Betriebssystemen den Wert der Registrierdatenbankvariablen **DB2INSTDEF**.

Vorgehensweise

Gehen Sie wie folgt vor, um die aktuelle Instanz zu ermitteln:

Prüfen Sie den Wert der relevanten Umgebungsvariablen.

Option	Bezeichnung
Zeigen Sie den Wert der Umgebungsvariablen DB2INSTANCE für die aktuelle Sitzung an.	Setzen Sie den folgenden Befehl ab: `db2 get instance`

Zeigen Sie den Wert der Systemumgebungsvariablen DB2INSTANCE an.	• Setzen Sie unter Windows-Betriebssystemen den folgenden Befehl ab:
	`echo %DB2INSTANCE%`
	• Setzen Sie unter Linux- und UNIX-Betriebssystemen den folgenden Befehl ab:
	`echo $DB2INSTANCE`

Zeigen Sie den Wert der Registrierdatenbankvariablen DB2INSTDEF an.	Setzen Sie den folgenden Befehl ab: `db2set DB2INSTDEF`

Definieren von Variablen auf Instanzebene in einer Umgebung mit partitionierten Datenbanken

In einer Umgebung mit partitionierten Datenbanken hängt die Vorgehensweise beim Definieren von Registrierdatenbankvariablen in der Profilregistrierdatenbank auf Instanzebene vom verwendeten Betriebssystem ab.

Informationen zu diesem Vorgang

Unter Linux- und UNIX-Betriebssystemen wird die Profilregistrierdatenbank auf Instanzebene im Verzeichnis `sqllib` in einer Textdatei gespeichert. Da das Verzeichnis `sqllib` in einem Dateisystem gespeichert ist, das von allen physischen Datenbankpartitionen gemeinsam genutzt wird, können Sie von jeder beliebigen Datenbankpartition aus Registrierdatenbankvariablen definieren.

Unter Windows-Betriebssystemen speichert der DB2-Datenbankmanager die Profilregistrierdatenbank auf Instanzebene in der Windows-Registrierdatenbank. Daher werden Daten nicht in allen physischen Datenbankpartitionen gemeinsam genutzt. Zum Definieren einer Variablen für alle Datenbankpartitionen müssen Sie den Befehl **rah** verwenden. Damit stellen Sie sicher, dass der zum Definieren der Variablen verwendete Befehl auf allen Computern ausgeführt wird. Wenn Sie eine Registrierdatenbankvariable von einer Datenbankpartition aus definieren und dazu nicht den Befehl **rah** verwenden, wird die Variable nur für die betreffende Datenbankpartition in der aktuellen Instanz definiert.

Sie können auch die Registrierdatenbankvariable **DB2REMOTEPREG** verwenden, um einen Computer, der nicht der Instanzeigner ist, so zu konfigurieren, dass er die Werte von Registrierdatenbankvariablen des Computers verwendet, der Instanzeigner ist.

Vorgehensweise

Gehen Sie wie folgt vor, um eine Registrierdatenbankvariable für alle Datenbankpartitionen der aktuellen Instanz zu definieren:

Setzen Sie den Befehl für das verwendete Betriebssystem von einer beliebigen Datenbankpartition aus ab.

- Setzen Sie unter Linux- und UNIX-Betriebssystemen den folgenden Befehl ab:
  ```
  db2set name_der_registrierdatenbankvariablen=neuer_wert
  ```
- Setzen Sie unter Windows-Betriebssystemen den folgenden Befehl ab:
  ```
  rah db2set name_der_registrierdatenbankvariablen=neuer_wert
  ```

Kumulative Registrierdatenbankvariablen

Mit einer kumulativen Registrierdatenbankvariablen können mehrere Registrierdatenbankvariablen zu einer Konfiguration zusammengefasst werden, die durch einen anderen Registrierdatenbankvariablennamen identifiziert wird. Jeder Registrierdatenbankvariablen, die zu dieser Gruppe gehört, ist eine vordefinierte Einstellung zugeordnet. Der kumulativen Registrierdatenbankvariablen wird ein Wert zugeordnet, der als Deklaration mehrerer Registrierdatenbankvariablen interpretiert wird.

Zweck einer kumulativen Registrierdatenbankvariablen ist die Vereinfachung der Registrierdatenbankkonfiguration für größere Betriebszielsetzungen.

Die einzige gültige kumulative Registrierdatenbankvariable ist **DB2_WORKLOAD**.

Gültige Werte für diese Variable sind:

- 1C
- CM
- COGNOS_CS
- FILENET_CM
- INFOR_ERP_LN
- MAXIMO
- MDM
- SAP
- TPM
- WAS
- WC
- WP

Jede Registrierdatenbankvariable, die implizit über eine kumulative Registrierdatenbankvariable konfiguriert ist, kann außerdem explizit definiert werden. Die explizite Definition einer Registrierdatenbankvariablen, der zuvor ein Wert über eine kumulative Registrierdatenbankvariable zugeordnet wurde, ist bei der Durchführung eines Leistungs- oder Diagnosetests nützlich. Die explizite Definition einer Variablen, die durch eine kumulative Registrierdatenbankvariable implizit konfiguriert wurde, wird als Überschreiben der Variablen bezeichnet.

Wenn Sie versuchen, eine explizit definierte Registrierdatenbankvariable durch eine kumulative Registrierdatenbankvariable zu ändern, wird eine Warnung ausgegeben und der explizit definierte Wert beibehalten. In dieser Warnung werden Sie informiert, dass der explizite Wert beibehalten wird. Wird zuerst die kumulative Registrierdatenbankvariable verwendet und anschließend die explizite Registrierdatenbankvariable angegeben, erhalten Sie keine Warnung.

Wenn Sie die kumulative Registrierdatenbankvariable abfragen, wird nur der dieser Variablen zugeordnete Wert angezeigt. Für die meisten Benutzer sind die Werte für die einzelnen Variablen nicht von Interesse.

Im folgenden Beispiel wird die Interaktion zwischen der Verwendung der kumulativen Registrierdatenbankvariablen und der expliziten Definition einer Registrierdatenbankvariablen dargestellt. Zum Beispiel können Sie zum Steuern der Datenbankumgebung die kumulative Registrierdatenbankvariable **DB2_WORKLOAD** auf den

Wert SAP setzen und die Registrierdatenbankvariable **DB2_SKIPDELETED** mit dem Wert NO überschreiben. Wenn Sie den Befehl **db2set** ausführen, werden folgende Ergebnisse angezeigt:

```
DB2_WORKLOAD=SAP
DB2_SKIPDELETED=NO
```

In einem weiteren Beispiel definieren Sie möglicherweise **DB2ENVLIST**, setzen die kumulative Registrierdatenbankvariable **DB2_WORKLOAD** auf den Wert SAP und überschreiben die Registrierdatenbankvariable **DB2_SKIPDELETED** mit dem Wert NO. Wenn Sie dann den Befehl **db2set** absetzen, wird für die Registrierdatenbankvariablen, die durch Setzen der kumulativen Registrierdatenbankvariablen automatisch konfiguriert wurden, der Name der kumulativen Registrierdatenbankvariablen in eckigen Klammern neben dem zugehörigen Wert angezeigt. Für die Registrierdatenbankvariable **DB2_SKIPDELETED** wird der Wert NO mit der Angabe [O] daneben angezeigt.

Wenn Sie die zu **DB2_WORKLOAD** gehörige Konfiguration nicht mehr benötigen, löschen Sie die impliziten Werte aller Registrierdatenbankvariablen in der Gruppe, indem Sie den Wert der kumulativen Registrierdatenbankvariablen löschen. Verwenden Sie zum Löschen des Werts der Variablen **DB2_WORKLOAD** den folgenden Befehl:

```
db2set DB2_WORKLOAD=
```

Nach dem Löschen des Werts der kumulativen Registrierdatenbankvariablen **DB2_WORKLOAD** müssen Sie die Datenbank erneut starten. Nach dem Neustart der Datenbank sind die Registrierdatenbankvariablen, die implizit durch die kumulative Registrierdatenbankvariable konfiguriert wurden, nicht mehr wirksam.

Durch das Löschen des Werts einer kumulativen Registrierdatenbankvariablen wird nicht gleichzeitig auch der Wert für eine Registrierdatenbankvariable gelöscht, die explizit gesetzt wurde. Dabei spielt es keine Rolle, dass die Registrierdatenbankvariable Mitglied der Gruppendefinition ist, die gelöscht wurde. Die explizite Definition für die Registrierdatenbankvariable bleibt bestehen.

Sie können die Werte für die einzelnen Registrierdatenbankvariablen anzeigen, die Teil der kumulativen Registrierdatenbankvariablen **DB2_WORKLOAD** sind. Sie wollen zum Beispiel die Werte prüfen, die verwendet würden, wenn Sie die Variable **DB2_WORKLOAD** auf den Wert SAP setzen würden. Zum Ermitteln der Werte, die bei der Definition von **DB2_WORKLOAD**=SAP verwendet würden, führen Sie den Befehl `db2set -gd DB2_WORKLOAD=SAP` aus.

Registrierdatenbank- und Umgebungsvariablen von DB2

DB2-Datenbankprodukte stellen eine Reihe von Registrierdatenbankvariablen und Umgebungsvariablen bereit, die Sie kennen sollten, um das Produkt betriebsbereit zu machen.

Zum Anzeigen einer Liste aller unterstützten Registrierdatenbankvariablen führen Sie den folgenden Befehl aus:

```
db2set -lr
```

Sie müssen Werte für Registrierdatenbankvariablen definieren, die Sie aktualisieren wollen, bevor Sie den Befehl **db2start** ausführen.

In der folgenden Tabelle sind alle Registrierdatenbankvariablen nach Kategorie aufgelistet.

Tabelle 122. Registrierdatenbank- und Umgebungsvariablen - Übersicht. Die zweite Spalte wird in zwei Spalten aufgeteilt.

Variablenkategorie	Name der Registrierdatenbank- bzw. Umgebungsvariablen	
Allgemein	DB2ACCOUNT DB2BIDI DB2_CAPTURE_LOCKTIMEOUT DB2CODEPAGE DB2_COLLECT_TS_REC_INFO DB2_CONNRETRIES_INTERVAL DB2CONSOLECP DB2DBDFT DB2DISCOVERYTIME DB2_ENFORCE_MEMBER_SYNTAX DB2FODC DB2_FORCE_APP_ON_MAX_LOG	DB2GRAPHICUNICODESERVER DB2INCLUDE DB2INSTDEF DB2INSTOWNER DB2_LIC_STAT_SIZE DB2LOCALE DB2_MAX_CLIENT_CONNRETRIES DB2_OBJECT_TABLE_ENTRIES DB2_SYSTEM_MONITOR_SETTINGS DB2TERRITORY DB2_VIEW_REOPT_VALUES
Systemumgebung	DB2_ALTERNATE_GROUP_LOOKUP DB2CONNECT_ENABLE_EURO_CODEPAGE DB2CONNECT_IN_APP_PROCESS DB2_COPY_NAME DB2_CPU_BINDING DB2DBMSADDR DB2_DIAGPATH DB2DOMAINLIST DB2ENVLIST DB2INSTANCE DB2INSTPROF DB2LDAPSecurityConfig DB2LIBPATH DB2LOGINRESTRICTIONS	DB2NODE DB2OPTIONS DB2_PARALLEL_IO DB2PATH DB2_PMAP_COMPATIBILITY DB2PROCESSORS DB2RCMD_LEGACY_MODE DB2RESILIENCE DB2_RESTORE_GRANT_ADMIN_AUTHORITIES DB2SYSTEM DB2_UPDDBCFG_SINGLE_DBPARTITION DB2_USE_PAGE_CONTAINER_TAG DB2_WORKLOAD
Kommunikation	DB2CHECKCLIENTINTERVAL DB2COMM DB2FCMCOMM DB2_FORCE_NLS_CACHE DB2_PMODEL_SETTINGS DB2RSHCMD DB2RSHTIMEOUT	DB2SORCVBUF DB2SOSNDBUF DB2TCP_CLIENT_CONTIMEOUT DB2TCP_CLIENT_KEEPALIVE_TIMEOUT DB2TCP_CLIENT_RCVTIMEOUT DB2TCPCONNMGRS DB2TCP_SERVER_KEEPALIVE_TIMEOUT
Befehlszeile	DB2BQTIME DB2BQTRY DB2_CLP_EDITOR DB2_CLP_HISTSIZE	DB2_CLPPROMPT DB2IQTIME DB2RQTIME
Umgebung mit partitionierten Datenbanken	DB2CHGPWD_EEE DB2_FCM_SETTINGS DB2_FORCE_OFFLINE_ADD_PARTITION	DB2_NUM_FAILOVER_NODES DB2_PARTITIONEDLOAD_DEFAULT DB2PORTRANGE
DB2 pureScale-Umgebung	DB2_DATABASE_CF_MEMORY	DB2_MCR_RECOVERY_PARALLELISM_CAP
Abfragecompiler	DB2_ANTIJOIN DB2_DEFERRED_PREPARE_SEMANTICS DB2_INLIST_TO_NLJN DB2_LIKE_VARCHAR DB2_MINIMIZE_LISTPREFETCH	DB2_NEW_CORR_SQ_FF DB2_OPT_MAX_TEMP_SIZE DB2_REDUCED_OPTIMIZATION DB2_SELECTIVITY DB2_SQLROUTINE_PREPOPTS

Tabelle 122. Registrierdatenbank- und Umgebungsvariablen - Übersicht (Forts.). Die zweite Spalte wird in zwei Spalten aufgeteilt.

Variablen-kategorie	Name der Registrierdatenbank- bzw. Umgebungsvariablen	
Leistung	DB2_ALLOCATION_SIZE DB2_APM_PERFORMANCE DB2ASSUMEUPDATE DB2_AVOID_PREFETCH DB2BPVARS DB2CHKPTR DB2CHKSQLDA DB2_EVALUNCOMMITTED DB2_EXTENDED_IO_FEATURES DB2_EXTENDED_OPTIMIZATION DB2_IO_PRIORITY_SETTING DB2_KEEP_AS_AND_DMS_CONTAINERS_OPEN DB2_KEEPTABLELOCK DB2_LARGE_PAGE_MEM DB2_LOGGER_NON_BUFFERED_IO DB2MAXFSCRSEARCH DB2_MAX_INACT_STMTS DB2_MAX_NON_TABLE_LOCKS DB2_MDC_ROLLOUT DB2MEMDISCLAIM DB2_MEM_TUNING_RANGE DB2_MMAP_READ	DB2_MMAP_WRITE DB2_NO_FORK_CHECK DB2NTMEMSIZE DB2NTNOCACHE DB2NTPRICLASS DB2NTWORKSET DB2_OVERRIDE_BPF DB2_PINNED_BP DB2PRIORITIES DB2_RCT_FEATURES DB2_RESOURCE_POLICY DB2_SELUDI_COMM_BUFFER DB2_SET_MAX_CONTAINER_SIZE DB2_SKIPDELETED DB2_SKIPINSERTED DB2_SMS_TRUNC_TMPTABLE_THRESH DB2_SORT_AFTER_TQ DB2_SQLWORKSPACE_CACHE DB2_TRUSTED_BINDIN DB2_USE_ALTERNATE_PAGE_CLEANING DB2_USE_FAST_PREALLOCATION DB2_USE_IOCP
Verschiedene Variablen	DB2ADMINSERVER DB2_ATS_ENABLE DB2AUTH DB2CLIINIPATH DB2_COMMIT_ON_EXIT DB2_COMMON_APP_DATA_PATH DB2_COMPATIBILITY_VECTOR DB2_CREATE_DB_ON_PATH DB2_DDL_SOFT_INVAL DB2_DISABLE_FLUSH_LOG DB2_DISPATCHER_PEEKTIMEOUT DB2_DJ_INI DB2_DMU_DEFAULT DB2_DOCHOST DB2_DOCPORT DB2DSDRIVER_CFG_PATH DB2DSDRIVER_CLIENT_HOSTNAME DB2_ENABLE_AUTOCONFIG_DEFAULT DB2_ENABLE_LDAP DB2_EVMON_EVENT_LIST_SIZE DB2_EVMON_STMT_FILTER DB2_EXTSECURITY DB2_FALLBACK DB2_FMP_COMM_HEAPSZ DB2_GRP_LOOKUP DB2_HADR_BUF_SIZE DB2_HADR_NO_IP_CHECK DB2_HADR_PEER_WAIT_LIMIT DB2_HADR_ROS DB2_HADR_SORCVBUF DB2_HADR_SOSNDBUF	DB2_INDEX_PCTFREE_DEFAULT DB2LDAP_BASEDN DB2LDAPCACHE DB2LDAP_CLIENT_PROVIDER DB2LDAPHOST DB2LDAP_KEEP_CONNECTION DB2LDAP_SEARCH_SCOPE DB2_LOAD_COPY_NO_OVERRIDE DB2_LIMIT_FENCED_GROUP DB2LOADREC DB2LOCK_TO_RB DB2_MAX_LOB_BLOCK_SIZE DB2_MEMORY_PROTECT DB2_MIN_IDLE_RESOURCES DB2_NCHAR_SUPPORT DB2NOEXITLIST DB2_NUM_CKPW_DAEMONS DB2_OPTSTATS_LOG DB2REMOTEPREG DB2_RESOLVE_CALL_CONFLICT DB2SATELLITEID DB2_SERVER_CONTIMEOUT DB2_SERVER_ENCALG DB2SORT DB2_STANDBY_ISO DB2STMM DB2_TRUNCATE_REUSESTORAGE DB2_UTIL_MSGPATH DB2_XBSA_LIBRARY DB2_XSLT_ALLOWED_PATH

Allgemeine Registrierdatenbankvariablen

DB2ACCOUNT

- Betriebssystem: Alle
- Standardwert: NULL
- Diese Variable definiert die Abrechnungszeichenfolge, die an den fernen Host gesendet wird. Einzelheiten siehe DB2 Connect - Benutzerhandbuch.

DB2BIDI

- Betriebssystem: Alle
- Standardwert: NO, Werte: YES oder NO
- Diese Variable aktiviert die bidirektionale Unterstützung von Sprachen, und die Variable **DB2CODEPAGE** dient zur Deklaration der zu verwendenden Codepage.

DB2_CAPTURE_LOCKTIMEOUT

- Betriebssystem: Alle
- Standardwert: NULL, Werte: ON oder NULL
- Diese Variable gibt an, dass beschreibende Informationen zu Sperrzeitlimitüberschreitungen zu dem Zeitpunkt zu protokollieren sind, zu dem sie auftreten. Folgende Informationen werden protokolliert: Die Hauptanwendungen, die an dem Sperrenkonflikt beteiligt sind, der zu der Sperrzeitlimitüberschreitung führte, die Details darüber, was diese Anwendungen ausführten, als die Überschreitung des Sperrzeitlimits auftrat, sowie die Details über die Sperre, die den Zugriffskonflikt verursacht hat. Die Informationen werden über den Anforderer der Sperre (die Anwendung, die den Zeitlimitfehler empfangen hat) und den aktuellen Sperreneigner erfasst. Zu jeder Sperrzeitlimitüberschreitung wird ein Textbericht geschrieben und in einer Datei gespeichert.

  Die Dateien werden mit der folgenden Namenskonvention erstellt: db2locktimeout.*par*.*AGENTID*.*jjjj-mm-tt-hh-mm-ss*. Dabei ist *par* die Datenbankpartitionsnummer, *AGENTID* ist die Agenten-ID und *jjjj-mm-tt-hh-mm-ss* ist die Zeitmarke, die aus Jahr, Monat, Tag, Stunde, Minuten und Sekunden besteht. In einer nicht partitionierten Datenbankumgebung hat *par* den Wert 0.

  Die Position der Datei hängt von dem Wert ab, der im Datenbankkonfigurationsparameter **diagpath** angegeben ist. Wenn der Parameter **diagpath** nicht definiert ist, befindet sich die Datei in einem der folgenden Verzeichnisse:

  – In Windows-Umgebungen:
    - Wenn Sie die Umgebungsvariable **DB2INSTPROF** nicht definiert haben, werden die Informationen in das folgende Verzeichnis geschrieben: *x*:*SQLLIB**DB2INSTANCE*. Dabei ist *x* die Laufwerkangabe, *SQLLIB* ist das Verzeichnis, das Sie für die Registrierdatenbankvariable **DB2PATH** angegeben haben, und *DB2INSTANCE* ist der Name des Instanzeigners.
    - Wenn Sie die Umgebungsvariable **DB2INSTPROF** definieren, werden die Informationen in das folgende Verzeichnis geschrieben: *x*:*DB2INSTPROF**DB2INSTANCE*. Dabei ist *x* die Laufwerkangabe, *DB2INSTPROF* der Name des Instanzprofilverzeichnisses und *DB2INSTANCE* ist der Name des Instanzeigners.

- Wenn Sie für die Umgebungsvariable **DB2INSTPROF** eine neue Position definieren, müssen Sie sicherstellen, dass an dieser Position die entsprechenden Dateien und Ordner zur Ausführung der Instanz vorhanden sind. Hierfür müssen Sie möglicherweise alle Dateien und Ordner von der vorhergehenden Position an die neue Position kopieren.

– In Linux- und UNIX-Umgebungen: Informationen werden in das Verzeichnis *INSTHOME*/sqllib/db2dump geschrieben. Dabei ist *INSTHOME* das Ausgangsverzeichnis der Instanz.

Löschen Sie Berichtsdateien zu Sperrzeitüberschreitungen, wenn Sie sie nicht mehr benötigen. Da diese Berichtsdateien an derselben Position wie andere Diagnoseprotokolle gespeichert werden, könnte das DB2-System herunterfahren, wenn sich das Verzeichnis vollständig füllt. Wenn Sie einige Berichtsdateien zu Sperrzeitüberschreitungen behalten wollen, versetzen Sie sie in ein anderes Verzeichnis (bzw. einen anderen Ordner) als das, in dem die DB2-Protokolle gespeichert werden.

- Änderungen an dieser Variable werden sofort wirksam und gelten für alle zukünftigen kompilierten SQL-Anweisungen. Es ist nicht nötig, die Instanz erneut zu starten oder den Befehl **db2set** mit dem Parameter **-immediate** abzusetzen.

Wichtig: Diese Variable ist veraltet und wird in einem zukünftigen Release möglicherweise entfernt, da neue Methoden zur Erfassung von Sperrzeitüberschreitungsereignissen unter Verwendung der Anweisung CREATE EVENT MONITOR FOR LOCKING zur Verfügung stehen.

DB2CODEPAGE

- Betriebssystem: Alle
- Standardwert: abgeleitet aus der vom Betriebssystem angegebenen Sprachenkennung
- Diese Variable gibt die Codepage der Daten an, die an DB2 für Datenbankclientanwendungen übergeben werden. Der Benutzer sollte **DB2CODEPAGE** nicht definieren, sofern dies nicht explizit in DB2-Dokumenten angegeben ist oder er vom DB2-Service dazu angeleitet wird. Wird **DB2CODEPAGE** auf einen Wert gesetzt, der vom Betriebssystem nicht unterstützt wird, können unerwartete Ergebnisse auftreten. Im Normalfall brauchen Sie **DB2CODEPAGE** nicht zu definieren, weil DB2 die Informationen zur Codepage automatisch aus dem Betriebssystem abruft.

  Anmerkung: Da Windows keine Unicode-Codepage (in den Ländereinstellungen von Windows) anstelle der ANSI-Codepage meldet, verhält sich eine Windows-Anwendung nicht wie ein Unicode-Client. Um dieses Verhalten außer Kraft zu setzen, setzen Sie die Registrierdatenbankvariable **DB2CODEPAGE** auf 1208 (für die Unicode-Codepage), damit die Anwendung sich wie eine Unicode-Anwendung verhält.

DB2_COLLECT_TS_REC_INFO

- Betriebssystem: Alle
- Standardwert: ON, Werte: ON oder OFF
- Diese Variable gibt an, ob DB2 alle Protokolldateien bei einer aktualisierenden Recovery eines Tabellenbereichs verarbeitet, unabhängig davon, ob die Protokolldateien Protokollsätze enthalten, die den Tabellenbereich betreffen. Wenn die Protokolldateien übersprungen werden sollen, die bekanntermaßen keine Protokollsätze für diesen Tabellenbereich enthalten, setzen Sie diese Variable auf den Wert ON. **DB2_COLLECT_TS_REC_INFO**

muss definiert werden, bevor die Protokolldateien erstellt und verwendet werden, sodass die Informationen, die zum Überspringen von Protokolldateien erforderlich sind, erfasst werden.

DB2_CONNRETRIES_INTERVAL

- Betriebssystem: Alle
- Standardwert: nicht definiert, Werte: ganzzahlige Werte in Sekunden
- Die Variable gibt die Ruhezeit (in Sekunden) zwischen aufeinanderfolgenden Verbindungsversuchen für die automatische Clientweiterleitungsfunktion an. Sie können diese Variable in Verbindung mit **DB2_MAX_CLIENT CONNRETRIES** verwenden, um das Wiederholungsverhalten für die automatische Clientweiterleitung zu konfigurieren.

  Wenn **DB2_MAX_CLIENT_CONNRETRIES** definiert ist, jedoch **DB2_CONNRETRIES_INTERVAL** nicht, nimmt **DB2_CONNRETRIES_INTERVAL** den Standardwert 30 an. Wenn **DB2_MAX_CLIENT_CONNRETRIES** nicht definiert ist, jedoch **DB2_CONNRETRIES_INTERVAL** definiert ist, gilt für **DB2_MAX_CLIENT_CONNRETRIES** der Standardwert 10. Wenn weder **DB2_MAX_CLIENT_CONNRETRIES** noch **DB2_CONNRETRIES_INTERVAL** definiert ist, kehrt die Funktion der automatischen Clientweiterleitung zur Standardfunktionsweise zurück. Das heißt, sie versucht die Verbindung zu einer Datenbank über eine Dauer von bis zu 10 Minuten wiederholt herzustellen.

DB2CONSOLECP

- Betriebssystem: Windows
- Standardwert: NULL, Werte: alle gültigen Codepagewerte
- Gibt die Codepage für die Anzeige von DB2- Nachrichtentext an. Wenn angegeben, überschreibt dieser Wert die Einstellung für die Codepage des Betriebssystems.

DB2DBDFT

- Betriebssystem: Alle
- Standardwert: NULL
- Diese Variable gibt den Aliasnamen der Datenbank an, die für implizite Verbindungen zu verwenden ist. Wenn eine Anwendung keine Datenbankverbindung hat, aber SQL- oder XQuery-Anweisungen abgesetzt werden, wird eine implizite Verbindung hergestellt, wenn die Umgebungsvariable **DB2DBDFT** mit einer Standarddatenbank definiert wurde.

DB2DISCOVERYTIME

- Betriebssystem: Windows
- Standardwert: 40 Sekunden, Mindestwert: 20 Sekunden
- Diese Variable gibt die Zeitdauer an, die der Discovery-Prozess SEARCH nach DB2-Systemen sucht.

DB2_ENFORCE_MEMBER_SYNTAX

- Betriebssystem: Alle
- Standardwert: OFF, Werte: OFF oder ON
- Mit dieser Variablen können Sie steuern, ob die Syntax für SQL-Anweisungen, DB2-Befehle und APIs auf die richtige Verwendung der Datenbankpartitionsschlüsselwörter überprüft wird, um festzustellen, ob stattdessen das Schlüsselwort MEMBER verwendet werden muss. In einer DB2 pureScale-Umgebung wird die Verwendung datenbankpartitionsspezifischer Schlüsselwörter wie z. B. DBPARTITIONNUM oder DATA-

BASE PARTITION standardmäßig toleriert, auch wenn sich die Operation auf ein DB2 Member bezieht. Wenn **DB2_ENFORCE_MEMBER_SYNTAX** aber auf ON eingestellt wird, muss das Schlüsselwort MEMBER richtig angegeben werden. Andernfalls wird SQL1538N zurückgegeben. Die Einstellung dieser Variablen wird ignoriert und hat außerhalb der DB2 pureScale-Umgebung keine Auswirkung.

DB2_EXPRESSION_RULES

- Betriebssystem: Alle
- Standardwert: Leer, Werte: RAISE_ERROR_PERMIT_SKIP oder RAISE_ERROR_PERMIT_DROP
- Die Einstellungen für die Registrierdatenbankvariable **DB2_EXPRESSION_RULES** steuern, wie das DB2-Optimierungsprogramm den Zugriffsplan für Abfragen bestimmt, die die Funktion RAISE_ERROR enthalten. Das Standardverhalten der Funktion RAISE_ERROR besteht darin, dass keine Filterung über den Ausdruck hinaus, der diese Funktion enthält, verschoben werden kann. Dies kann dazu führen, dass keine Vergleichselemente während Tabellenzugriffen angewendet werden, was einen übermäßigen Berechnungsaufwand für Ausdrücke, übermäßiges Sperren und eine schlechte Abfrageleistung zur Folge haben kann.

  In bestimmten Fällen ist dieses Verhalten zu strikt. Je nach den jeweiligen Geschäftsanforderungen der Anwendung spielt es möglicherweise keine Rolle, ob Vergleichselemente und Joins angewendet werden, bevor die Funktion RAISE_ERROR angewendet wird. Im Kontext einer Sicherheitsimplementierung auf Zeilenebene ist zum Beispiel in der Regel ein Ausdruck der folgenden Form vorhanden:

  ```
  CASE WHEN <bedingungen zur prüfung des zugriffs auf diese zeile>
       THEN NULL
       ELSE RAISE_ERROR(...)
  END
  ```

  Für die Anwendung ist es vielleicht nur wichtig, den Zugriff auf die Zeilen zu prüfen, die von der Abfrage ausgewählt werden, und nicht den Zugriff auf jede Zeile in der Tabelle zu prüfen. Daher könnten Vergleichselemente beim Zugriff auf die Basistabelle angewendet werden und der Ausdruck, der die Funktion RAISE_ERROR enthält, muss nur ausgeführt werden, nachdem sämtliche Filterungen erfolgt sind. In diesem Fall kann **DB2_EXPRESSION_RULES**=RAISE_ERROR_PERMIT_SKIP ein geeigneter Wert sein.

  Eine weitere Alternative ergibt sich im Kontext der Sicherheit auf Spaltenebene (COLUMN LEVEL). In diesem Fall sind in der Regel Ausdrücke der folgenden Form vorhanden:

  ```
  CASE WHEN <bedingungen zur prüfung des zugriffs auf diese zeile und spalte>
       THEN <tabelle.spalte>
       ELSE RAISE_ERROR(...)
  END
  ```

  In diesem Fall bezweckt die Anwendung vielleicht, dass Fehler nur ausgelöst werden sollen, wenn der Benutzer versucht, die Daten für eine bestimmte Zeile zu empfangen, und eine bestimmte Spalte einen Wert enthält, den der Benutzer nicht abrufen darf. In diesem Fall führt die Einstellung **DB2_EXPRESSION_RULES**=RAISE_ERROR_PERMIT_DROP nur dazu, dass die Anweisung mit der Funktion RAISE_ERROR ausgewertet wird,

wenn die betreffende Spalte von einem Vergleichselement oder einer Spaltenfunktion verwendet oder als Ausgabe zu einer Abfrage zurückgegeben wird.

DB2FODC

- Betriebssystem: Alle
- Standardwert: die Verkettung aller FODC-Parameter (siehe folgende Liste)
  - Für Linux und UNIX: "CORELIMIT=*wert* DUMPCORE=ON DUMPDIR=*diagpath*"
  - Für Windows: "DUMPDIR=*diagpath*"

  Beachten Sie, dass die Parameter durch Leerzeichen getrennt angegeben werden.
- Diese Registrierdatenbankvariable steuert eine Gruppe von Fehlerdiagnoseparametern, die bei der First Occurrence Data Collection (FODC - Datenerfassung beim ersten Auftreten) verwendet werden. Mit der Variablen **DB2FODC** können verschiedene Aspekte der Datenerfassung in Ausfallsituationen gesteuert werden.

  Diese Registrierdatenbankvariable wird einmal während des Starts der DB2-Instanz gelesen. Wenn Sie die FODC-Parameter online aktualisieren wollen, verwenden Sie das Tool **db2pdcfg**. Mit der Registrierdatenbankvariablen **DB2FODC** können Sie die Konfiguration über Neustarts hinweg beibehalten. Es müssen weder alle Parameter angegeben werden noch müssen sie in einer bestimmten Reihenfolge angegeben werden. Jedem Parameter, der nicht angegeben wird, wird der Standardwert zugewiesen. Wenn Sie zum Beispiel keinen Auszug der Kerndateien erstellen wollen, während sich die anderen Parameter standardmäßig verhalten sollen, geben Sie den folgenden Befehl ein:

  ```
  db2set DB2FODC="DUMPCORE=OFF"
  ```

  Parameter:

  CORELIMIT

  - Betriebssystem: Linux und UNIX
  - Standardwert: Aktueller Wert für 'ulimit', Werte `0` bis `unlimited`
  - Diese Option gibt die maximale Größe (in Byte) der Kerndateien an, die erstellt werden. Dieser Wert überschreibt die aktuelle Einstellung für die Größenbegrenzung von Kerndateien. Ein besonderes Augenmerk sollte dem verfügbaren Dateisystemspeicher gelten, da Kerndateien recht groß sein können. Die Größe hängt von der DB2-Konfiguration und dem Status des Prozesses zum Zeitpunkt des Auftretens des Problems ab.

    Wenn **CORELIMIT** definiert ist, verwendet DB2 diesen Wert, um die aktuelle Einstellung für die maximale Größe von Kerndateien für Benutzer (ulimit) zum Generieren der Kerndatei zu überschreiben.

    Wenn **CORELIMIT** nicht angegeben wird, setzt DB2 die Kerndateigröße auf den Wert, der der aktuellen ulimit-Einstellung entspricht.

    Anmerkung: Alle Änderungen an der Größenbegrenzung von Kerndateien für Benutzer bzw. an **CORELIMIT** werden erst nach dem nächsten Neustart der DB2-Instanz wirksam.

COS

– Betriebssystem: Alle
– Standardwert: ON, Werte: ON oder OFF
– Diese Option gibt an, ob das Script **db2cos** aktiviert ist. Mit diesem Parameter können die folgenden Parameter verwendet werden:

  COS_SLEEP

  - Standardwert: 3, Werte: 0 bis unbegrenzt
  - Diese Option gibt die Ruhezeit in Sekunden zwischen dem Prüfen der Größe der generierten Ausgabedatei an.

  COS_TIMEOUT

  - Standardwert: 30, Werte: 0 bis unbegrenzt
  - Diese Option gibt die Wartezeit in Sekunden vor dem Beenden des Scripts an.

  COS_COUNT

  - Standardwert: 255, Werte: 0 bis 255
  - Diese Option gibt an, wie oft **db2cos** während einem Datenbankmanager-Trap ausgeführt werden soll.

  COS_SQLO_SIG_DUMP

  - Standardwert: ON, Werte: ON oder OFF
  - Diese Option gibt an, ob **db2cos** aktiviert wird, wenn das Signal **SQLO_SIG_DUMP** empfangen wird.

DUMPCORE

– Betriebssystem: Linux, Solaris, AIX
– Standardwert: AUTO, Werte: AUTO, ON oder OFF
– Diese Option gibt an, ob die Generierung von Kerndateien erfolgen soll. Kerndateien, die zur Fehlerbestimmung verwendet und in dem durch **diagpath** angegebenen Verzeichnis erstellt werden, enthalten das gesamte Prozessimage des DB2-Prozesses, der beendet wird. Ob allerdings tatsächlich ein Kerndateispeicherauszug erfolgt, hängt von der aktuellen Einstellung von 'ulimit' und dem Wert des Parameters **CORELIMIT** ab. Einige Betriebssysteme besitzen auch Konfigurationseinstellungen für Kernspeicherauszüge, die das Verhalten von Anwendungsspeicherauszügen möglicherweise vorgeben. Die Einstellung AUTO bewirkt, dass eine Kerndatei in dem Fall generiert wird, dass ein Trap nicht toleriert werden kann, wenn die Registrierdatenbankvariable **DB2RESILIENCE** auf den Wert ON gesetzt ist. Bei der Einstellung **DUMPCORE**=ON wird immer eine Kerndatei generiert, indem die Einstellung der Registrierdatenbankvariablen **DB2RESILIENCE** außer Kraft gesetzt wird.

  Die empfohlene Methode zur Inaktivierung von Kerndateispeicherauszügen besteht darin, den Parameter **DUMPCORE** auf OFF zu setzen.

DUMPDIR

– Betriebssystem: Alle

– Standardwert: **diagpath**-Verzeichnis oder das Standarddiagnoseverzeichnis, wenn **diagpath** nicht definiert ist, Werte: *pfad_zu_verzeichnis*
– Diese Option gibt den absoluten Pfadnamen des Verzeichnisses für die Erstellung von Kerndateien an.

FODCPATH

– Betriebssystem: Alle
– Standardwert: Pfad, der über den Konfigurationsparameter **DIAGPATH** des Datenbankmanagers definiert wird, Werte: *fodc-pfadname*
– Diese Option gibt den absoluten Pfadnamen des Verzeichnisses an, in das das FODC-Paket geleitet werden soll. Bei der Variablen *fodc-pfadname* muss es sich um ein bereits vorhandenes Verzeichnis handeln, für das Schreibzugriff durch die Teilkomponenten, für die es gesetzt ist, und für die fmp-Prozesse besteht, die für diese Teilkomponenten ausgeführt werden.

SERVICELEVEL

– Betriebssystem: Alle
– Standardwert: ulimit-Einstellung `AUTOMATIC`; Werte: `AUTOMATIC`, `BASIC` oder `FULL`
– Diese Option gibt an, wie Daten in Panic-Situationen, bei Traps und Fehlern erfasst werden, die auf fehlerhafte Daten hinweisen können. DB2 generiert Diagnoseinformationen für den Konfigurations- und Problemkontext. Wenn z. B. ein Trap toleriert werden kann, werden nur die dringendsten Diagnoseinformationen generiert, um einen Rollback der Transaktion durchzuführen und der Anwendung möglichst bald zu antworten, sodass Ressourcen freigegeben werden, auf die andere Anwendungen warten. Wenn ein Trap nicht toleriert werden kann, können Diagnoseinformationen (wie z. B. db2cos-Datenerfassungsscripts und Kernspeicherauszüge) zugunsten der Verfügbarkeit in DB2 pureScale-Konfigurationen begrenzt werden. Das Standardverhalten für die Generierung von Diagnoseinformationen ist die SERVICELEVEL-Einstellung AUTOMATIC.

  Folgende Optionen werden für diesen Parameter unterstützt:

  AUTOMATIC
  Diese Einstellung gibt an, dass die gültige SERVICELEVEL-Einstellung (d. h. `BASIC` oder `FULL`) für die Member während der Laufzeit und für die CF beim Start gewählt werden muss. Im Moment wird die Option `BASIC` nur für DB2 pureScale-Umgebungen gewählt, die mehrere Member enthalten.

  BASIC Diese SERVICELEVEL-Einstellung gibt an, dass ein Speicherauszug nur für eine minimale Menge von FODC-Daten erstellt werden soll. Die Verarbeitung von Kernspeicherauszügen wird inaktiviert, sodass der Speicherauszug auf den betroffenen Thread beschränkt wird, und die vollständige Verarbeitung der Aufrufscripts wird inaktiviert.

  FULL Diese SERVICELEVEL-Einstellung gibt an, dass ein

Speicherauszug für die maximale Menge von FODC-Daten erstellt werden soll. Dies umfasst Kernspeicherauszüge, die Auszüge aller zugehöriger Komponenten und den Aufruf der Aufrufscripts.

DB2_FORCE_APP_ON_MAX_LOG

- Betriebssystem: Alle
- Standardwert: TRUE, Werte: TRUE oder FALSE
- Gibt an, was geschieht, wenn der Wert des Konfigurationsparameters **max_log** überschritten wird. Bei TRUE wird die Anwendung zwangsweise von der Datenbank getrennt und die UOW (Unit of Work, Arbeitseinheit) mit ROLLBACK rückgängig gemacht.

  Bei FALSE schlägt die aktuelle Anweisung fehl. Die Anwendung kann weiterhin von vorherigen Anweisungen in der UOW fertiggestellte Arbeit mit COMMIT festschreiben, oder sie kann die ausgeführte Arbeit mit ROLLBACK rückgängig machen, um die UOW zurückzusetzen.

  Anmerkung: Diese DB2-Registrierdatenbankvariable beeinflusst die Fähigkeit des Importdienstprogramms zur Wiederherstellung nach Situationen, in denen der Platz im Protokoll nicht ausreicht. Wenn für **DB2_FORCE_APP_ON_MAX_LOG** der Wert TRUE definiert wird und Sie den Befehl **IMPORT** mit der Befehlsoption **COMMITCOUNT** eingeben, kann das Importdienstprogramm kein Commit durchführen, um Speicherknappheit im aktiven Protokoll zu vermeiden. Wenn das Importdienstprogramm den Fehler SQL0964C (Transaktionsprotokoll voll) feststellt, wird seine Verbindung zur Datenbank getrennt, und die aktuelle UOW wird rückgängig gemacht.
- Änderungen an dieser Variable werden sofort wirksam und gelten für alle zukünftigen kompilierten SQL-Anweisungen. Es ist nicht nötig, die Instanz erneut zu starten oder den Befehl **db2set** mit dem Parameter **-immediate** abzusetzen.

DB2GRAPHICUNICODESERVER

- Betriebssystem: Alle
- Standardwert: OFF, Werte: ON oder OFF
- Diese Registrierdatenbankvariable dient zur Zulassung vorhandener Anwendungen, die zum Einfügen von Grafikdaten in eine Unicode-Datenbank geschrieben sind. Ihre Verwendung ist nur für Anwendungen erforderlich, die sqldbchar-Daten (Grafikdaten) speziell in Unicode anstatt in der Codepage des Clients senden. ('sqldbchar' ist ein unterstützter SQL-Datentyp in C und C++, der ein einzelnes Doppelbytezeichen aufnehmen kann.) Wenn Sie diese Registrierdatenbankvariable auf den Wert ON setzen, teilen Sie der Datenbank mit, dass der Eingang von Unicode-Grafikdaten zu erwarten ist, und die Anwendung erwartet ebenfalls, Unicode-Grafikdaten zu empfangen.

DB2INCLUDE

- Betriebssystem: Alle
- Standardwert: Aktuelles Verzeichnis
- Gibt einen Pfad an, der bei der Verarbeitung der SQL-Anweisung INCLUDE textdatei während der Verarbeitung von **DB PREP** zu verwenden ist. Er gibt eine Liste von Verzeichnissen an, in denen sich die INCLUDE-Datei befinden könnte. Im Handbuch Developing Embedded SQL Applications finden Sie Beschreibungen, wie **DB2INCLUDE** in den verschiedenen vorkompilierten Sprachen verwendet wird.

DB2INSTDEF

- Betriebssystem: Windows
- Standardwert: DB2
- Diese Variable definiert den Wert, der zu verwenden ist, wenn **DB2INSTANCE** nicht definiert ist.

DB2INSTOWNER

- Betriebssystem: Windows
- Standardwert: **NULL**
- Die Registrierdatenbankvariable, die bei der ersten Erstellung der Instanz in der DB2-Profilregistrierdatenbank erstellt wird. Diese Variable wird auf den Namen der Instanzeignermaschine gesetzt.

DB2_LIC_STAT_SIZE

- Betriebssystem: Alle
- Standardwert: NULL, Bereich: 0 bis 32767
- Diese Variable legt die Maximalgröße (in MB) der Datei mit den Lizenzstatistikdaten für das System fest. Der Wert 0 schaltet das Sammeln von Lizenzstatistikdaten aus. Wenn die Variable nicht erkannt wird oder nicht definiert ist, wird standardmäßig eine uneingeschränkte Größe angenommen. Die Statistikdaten werden über die Lizenzzentrale angezeigt.

DB2LOCALE

- Betriebssystem: Alle
- Standardwert: NO, Werte: YES oder NO
- Diese Variable gibt an, ob das Standardlocale "C" eines Prozesses nach dem Aufrufen von DB2 auf das Standardlocale "C" zurückgesetzt wird und ob das Prozesslocale nach dem Aufrufen einer DB2-Funktion auf das ursprüngliche 'C' zurückzusetzen ist. Wenn das ursprüngliche Locale nicht 'C' war, wird diese Registrierdatenbankvariable ignoriert.

DB2_MAX_CLIENT_CONNRETRIES

- Betriebssystem: Alle
- Standardwert: nicht definiert, Werte: eine ganze Zahl für die maximale Anzahl von Wiederholungsversuchen zur Herstellung der Verbindung
- Diese Variable gibt die maximale Anzahl von Verbindungsversuchen an, die von der Funktion zur automatischen Clientweiterleitung unternommen werden sollen. Sie können diese Variable in Verbindung mit **DB2_CONNRETRIES_INTERVAL** verwenden, um das Wiederholungsverhalten für die automatische Clientweiterleitung zu konfigurieren.

  Wenn **DB2_MAX_CLIENT_CONNRETRIES** definiert ist, jedoch **DB2_CONNRETRIES_INTERVAL** nicht, nimmt **DB2_CONNRETRIES_INTERVAL** den Standardwert 30 an. Wenn **DB2_MAX_CLIENT_CONNRETRIES** nicht definiert ist, jedoch **DB2_CONNRETRIES_INTERVAL** definiert ist, gilt für **DB2_MAX_CLIENT_CONNRETRIES** der Standardwert 10. Wenn weder **DB2_MAX_CLIENT_CONNRETRIES** noch **DB2_CONNRETRIES_INTERVAL** definiert ist, kehrt die Funktion der automatischen Clientweiterleitung zur Standardfunktionsweise zurück. Das heißt, sie versucht die Verbindung zu einer Datenbank über eine Dauer von bis zu 10 Minuten wiederholt herzustellen.

DB2_OBJECT_TABLE_ENTRIES

- Betriebssystem: Alle
- Standardwert: 0, Werte: 0 bis 65532

Der tatsächliche Maximalwert, der auf Ihrem System möglich ist, hängt von der Seitengröße und der Speicherbereichsgröße (EXTENTSIZE) ab, kann jedoch den Wert 65.532 nicht überschreiten.

- Diese Variable gibt die erwartete Anzahl von Objekten in einem Tabellenbereich an. Wenn Sie wissen, dass in einem DMS-Tabellenbereich eine große Anzahl von Objekten (z. B. 1000 oder mehr) erstellt wird, sollten Sie diese Registrierdatenbankvariable auf die ungefähre Anzahl setzen, bevor Sie den Tabellenbereich erstellen. Dadurch wird zusammenhängender Speicherplatz für Objektmetadaten bei der Erstellung des Tabellenbereichs reserviert. Die Reservierung von Speicher verringert die Wahrscheinlichkeit, dass ein Online-Backup Operationen blockiert, die Einträge in den Metadaten aktualisieren (wie z. B. CREATE INDEX, **IMPORT REPLACE**). Sie erleichtert außerdem eine spätere Änderung der Größe des Tabellenbereichs, weil die Metadaten am Anfang des Tabellenbereichs gespeichert werden.

  Wenn die Anfangsgröße des Tabellenbereichs nicht ausreichend groß ist, um zusammenhängenden Speicher zu reservieren, wird die Erstellung des Tabellenbereichs fortgesetzt, ohne den zusätzlichen Speicher zu reservieren.

DB2_SYSTEM_MONITOR_SETTINGS

- Betriebssystem: Alle
- Änderungen an dieser Variable werden sofort wirksam und gelten für alle zukünftigen kompilierten SQL-Anweisungen. Es ist nicht nötig, die Instanz erneut zu starten oder den Befehl **db2set** mit dem Parameter **-immediate** abzusetzen.
- Diese Registrierdatenbankvariable steuert eine Gruppe von Parametern, mit denen Sie das Verhalten verschiedener Aspekte der DB2-Überwachung ändern können. Trennen Sie jeden Parameter durch ein Semikolon wie im folgenden Beispiel:

  ```
  db2set DB2_SYSTEM_MONITOR_SETTINGS=OLD_CPU_USAGE:TRUE;DISABLE_CPU_USAGE:TRUE
  ```

  Bei jeder Definition von **DB2_SYSTEM_MONITOR_SETTINGS** müssen die einzelnen Parameter explizit angegeben werden. Jeder Parameter, den Sie in dieser Variablen nicht angeben, wird auf seinen Standardwert zurückgesetzt. Betrachten Sie folgendes Beispiel:

  ```
  db2set DB2_SYSTEM_MONITOR_SETTINGS=DISABLE_CPU_USAGE:TRUE
  ```

  In diesem Fall wird OLD_CPU_USAGE auf die Standardeinstellung zurückgesetzt.

  Anmerkung: Derzeit verfügt diese Registrierdatenbankvariable nur über Einstellungen für Linux; zusätzliche Einstellungen für andere Betriebssysteme werden zukünftigen Releases hinzugefügt.

- Parameter:

  OLD_CPU_USAGE

  - Betriebssystem: Linux
  - Werte: TRUE/ON, FALSE/OFF
  - Standardwert unter RHEL4 und SLES9: TRUE (Hinweis: Die Einstellung FALSE für OLD_CPU_USAGE wird ignoriert, nur die alte Funktionsweise wird verwendet.)
  - Standardwert unter RHEL5, SLES10 und anderen: FALSE

– Dieser Parameter steuert, wie die Instanz CPU-Nutzungszeiten auf Linux-Plattformen abruft. Bei TRUE wird die ältere Methode zum Abruf von CPU-Nutzungszeiten verwendet. Diese Methode gibt CPU-Nutzungszeiten für Systemprozesse und Benutzerprozesse zurück, benötigt dazu jedoch mehr CPU-Zeit (d. h., sie ist mit größerem Systemaufwand verbunden). Bei FALSE wird die neuere Methode zum Abruf der CPU-Belastung verwendet. Diese Methode gibt nur den Wert der CPU-Belastung durch den Benutzer zurück, ist jedoch schneller, weil sie weniger Systemaufwand verursacht.

DISABLE_CPU_USAGE

– Betriebssystem: Linux
– Werte: TRUE/ON, FALSE/OFF
– Standardwert unter RHEL4 und SLES9: TRUE
– Standardwert unter RHEL5, SLES10 und anderen: FALSE
– Mit diesem Parameter können Sie festlegen, ob die CPU-Belastung gelesen wird oder nicht. Wenn DISABLE_CPU_USAGE aktiviert (TRUE) ist, wird die CPU-Belastung nicht gelesen, sodass Sie den Aufwand umgehen können, der manchmal mit dem Abrufen der CPU-Belastung verbunden ist.

DB2TERRITORY

- Betriebssystem: Alle
- Standardwert: abgeleitet aus der vom Betriebssystem angegebenen Sprachenkennung
- Diese Variable gibt den Regions- oder Gebietscode der Clientanwendung an, was sich auf die Formate für Datum und Uhrzeit auswirkt.

DB2_VIEW_REOPT_VALUES

- Betriebssystem: Alle
- Standardwert: NO, Werte: YES, NO
- Diese Variable gibt allen Benutzern die Möglichkeit, die im Cache abgelegten Werte einer reoptimierten SQL- oder XQuery-Anweisung in der Tabelle EXPLAIN_PREDICATE zu speichern, wenn die Anweisung mit EXPLAIN bearbeitet wird. Wenn diese Variable auf NO gesetzt ist, kann nur der DBADM diese Werte in der Tabelle EXPLAIN_PREDICATE speichern.

Systemumgebungsvariablen

DB2_ALTERNATE_GROUP_LOOKUP

- Betriebssystem: AIX, Linux
- Standardwert: NULL, Werte: NULL, GETGRSET unter AIX, GETGROUPLIST unter Linux
- Diese Variable ermöglicht es DB2-Datenbanksystemen, Gruppeninformationen aus einer alternativen Quelle abzurufen, die vom Betriebssystem bereitgestellt wird. Unter AIX wird die Funktion getgrset verwendet. Diese Funktion bietet die Möglichkeit, Gruppen aus einer anderen Position als lokalen Dateien über ladbare Authentifizierungsmodule (Loadable Authentication Modules) abzurufen.

DB2_CLP_EDITOR

Detaillierte Informationen siehe DB2_CLP_EDITOR in „Befehlszeilenvariablen".

DB2_CLP_HISTSIZE

Detaillierte Informationen siehe DB2_CLP_HISTSIZE in „Befehlszeilenvariablen".

DB2CONNECT_ENABLE_EURO_CODEPAGE

- Betriebssystem: Alle
- Standardwert: NO, Werte: YES oder NO
- Setzen Sie diese Variable auf YES auf allen DB2 Connect-Clients und -Servern, die eine Verbindung zu einem DB2 für z/OS-Server oder einem DB2 für IBM i-Server herstellen, auf denen die Unterstützung des Eurosymbols erforderlich ist. Wenn Sie diese Variable auf YES setzen, wird die aktuelle Anwendungscodepage der äquivalenten CCSID (ID des codierten Zeichensatzes) zugeordnet, die die Unterstützung für das Eurosymbol explizit angibt. Dadurch stellt DB2 Connect die Verbindung zu dem DB2 für z/OS-Server bzw. dem DB2 für IBM i-Server unter Verwendung einer CCSID her, die eine Obermenge der CCSID der aktuellen Anwendungscodepage ist und die zudem das Eurosymbol unterstützt. Wenn der Client zum Beispiel eine Codepage verwendet, die der CCSID 1252 zugeordnet wird, stellt der Client die Verbindung unter Verwendung der CCSID 5348 her.

DB2CONNECT_IN_APP_PROCESS

- Betriebssystem: Alle
- Standardwert: YES, Werte: YES oder NO
- Wenn Sie diese Variable auf den Wert NO setzen, werden lokale DB2 Connect-Clients auf einem DB2 Enterprise Server Edition-System gezwungen, innerhalb eines Agenten aktiv zu sein. Einige Vorteile der Ausführung innerhalb eines Agenten bestehen darin, dass lokale Clients auf diese Weise überwacht werden und die SYSPLEX-Unterstützung nutzen können.

DB2_COPY_NAME

- Betriebssystem: Windows
- Standardwert: Der Name der Standardkopie von DB2, die auf Ihrem System installiert ist. Werte: Der Name einer Kopie von DB2, die auf Ihrem System installiert ist. Der Name kann maximal 128 Zeichen lang sein.
- Die Variable **DB2_COPY_NAME** speichert den Namen der Kopie von DB2, die momentan verwendet wird. Wenn auf Ihrer Maschine mehrere DB2-Kopien installiert sind, können Sie mit **DB2_COPY_NAME** nicht zu einer anderen Kopie von DB2 wechseln. Sie müssen den Befehl *INSTALLPATH*\bin\db2envar.bat ausführen, um die zurzeit verwendete Kopie zu ändern, wobei *INSTALLPATH* die Position angibt, an der die DB2-Kopie installiert ist.

DB2_CPU_BINDING

- Betriebssystem: Linux
- Standardwert:
  - Wenn sich ein DB2-Member und eine Cluster-Caching-Funktion (CF) nicht auf demselben Host befinden:
    - Für das Member: NUM_CORES=max(1, floor(0.8*totalCores))
    - Für die Cluster-Caching-Funktion: NUM_CORES=totalCores - die zuvor angegebene Zahl.
  - Wenn ein DB2-Member und eine Cluster-Caching-Funktion nicht denselben Host nutzen, wird diese Variable nicht festgelegt.

- Diese Registrierdatenbankvariable steuert die CPU-Veknüpfung. Damit Änderungen an dieser Variablen wirksam werden, müssen Sie die DB2-Instanz erneut starten.

  Parameter:

  NUM_CORES
  - Betriebssystem: Linux
  - Standardwert: Wenn sich das Member oder die CF auf demselben Host befinden, werden DB2 etwa 80 % der insgesamt verfügbaren Prozessorkerne (Cores) zugeordnet und der Rest wird der CF zugeordnet. Werte: 0 < x < (Anzahl physischer Kerne auf dem Host)
  - Diese Option gibt die Anzahl Kerne an, mit denen die Prozesse des Members bzw. der CF verknüpft sind. Sie können `NUM_CORES` verwenden, um die Sub-Capacity-Lizenzierung des DB2-Produkts zu konfigurieren. Die Anzahl der Prozessorkerne kann eine ganze Zahl oder eine Bruchzahl sein, sodass Sie einen oder mehrere Hardware-Threads hinzufügen können, wenn simultanes Multithreading (SMT) aktiviert ist.

  PROCESSOR_LIST
  - Betriebssystem: Linux
  - Standardwert: nicht definiert, Werte: beliebige Prozessorzahl
  - Diese Option gibt an, an welche logischen Prozessoren DB2 gebunden wird. So erhalten Sie die vollständige Kontrolle über die Zahl der logischen Prozessoren (oder Kerne) und darüber, auf welchem CPU-Paket (bzw. welcher Socket) sie sich befinden werden. Wenn Sie versuchen, sowohl `PROCESSOR_LIST` als auch `NUM_CORES` mit **DB2_CPU_BINDING** zu definieren, wird `NUM_CORES` ignoriert.

  Aspekte des Light-Neustarts

  Wenn ein Member als Gastmember auf einem Host erneut gestartet wird, auf dem bereits ein Member ausgeführt wird, wird das Light-Neustart-Member mit den Kernen verknüpft, die bereits von dem vorhandenen Member verwendet werden (bis zur Anzahl der durch **DB2_CPU_BINDING** angegebenen Kerne). Wenn ein Member als Gastmember auf einem Host erneut gestartet wird, der über weniger Kerne als durch **DB2_CPU_BINDING** angegeben verfügt, wird das Member an die Anzahl Kerne auf dem Host gebunden.

  Jedes Mal, wenn Sie **DB2_CPU_BINDING** festlegen, wird jeder nicht explizit definierte Parameter im Profil auf Instanzebene gelöscht. Schließen Sie jeden Parameter und den zugehörigen Wert in Anführungszeichen ein, wie aus folgenden Beispielen hervorgeht.

  Beispiel 1

  Ein Benutzer möchte das erste Member (mit der ID 0) von DB2-Instanz db2inst1 mit einem Kern auf einer Hostmaschine mit zwei Kernen verknüpfen:

  ```
  db2set -i db2inst1 0 DB2_CPU_BINDING="NUM_CORES=1"
  ```

  Beispiel 2

  Ein Benutzer möchte alle Member in 'db2inst1' an fünf logische Prozessoren auf einer Hostmaschine mit acht Kernen und aktiviertem Intel HTT (d. h., es gibt 16 logische Prozessoren) binden:

```
db2set -i db2inst1 DB2_CPU_BINDING="NUM_CORES=2.5"
```

Beispiel 3

Ein Benutzer möchte angeben, an wie viele Kerne die primäre CF (mit ID 128) gebunden ist:

```
db2set -i db2inst1 128 DB2_CPU_BINDING="NUM_CORES=4"
```

Beispiel 4

Ein Benutzer möchte DB2 für db2inst1 auf Member 0 an eine bestimmte Gruppe von logischen Prozessoren binden:

```
db2set -i db2inst1 0 DB2_CPU_BINDING="PROCESSOR_LIST=2,10,6,14"
```

DB2DBMSADDR

- Betriebssystem: Linux auf x86, Linux auf zSeries (31-bit) und Windows 32-bit
- Standardwert: NULL unter Linux-Betriebssystemen, 0x20000000 unter Windows-Betriebssystemen, Werte: virtuelle Adressen aus dem Bereich von 0x09000000 bis 0xB0000000 in Inkrementen von 0x10000 unter Linux-Betriebssystemen, von 0x20000000 bis 0xB0000000 in Inkrementen von 0x10000 unter Windows-Betriebssystemen
- Die Registrierdatenbankvariable **DB2DBMSADDR** gibt die Standardadresse der Datenbank für gemeinsam genutzten Speicher im Hexadezimalformat an.

  Diese Variable kann zur Feinabstimmung der Adressraumbelegung von DB2-Prozessen verwendet werden. Diese Variable ändert die Position des gemeinsam genutzten Speichers für die Instanz von der aktuellen Position bei der virtuellen Adresse 0x10000000 in den neuen Wert.

  Anmerkung: Eine falsche Adresse kann schwerwiegende Probleme auf dem DB2-Datenbanksystem verursachen, die von einem Fehlschlagen des Starts einer DB2-Instanz bis hin zum Fehlschlagen der Verbindung zur Datenbank reichen können. Eine falsche Adresse ist eine Adresse, die einen Konflikt mit einem Adressbereich verursacht, der bereits verwendet wird oder der für andere Daten oder Programme reserviert ist. Dieses Problem können Sie beheben, indem Sie die Registrierdatenbankvariable **DB2DBMSADDR** mit dem folgenden Befehl auf NULL setzen:

  ```
  db2set DB2DBMSADDR=
  ```

  Anmerkung: Bevor Sie die Einstellung dieser Variablen ändern, müssen Sie die Instanz und alle DB2-Prozesse stoppen. Wenn die Instanz aktiv ist, während die Variable eingestellt wird, schlagen alle nachfolgenden **db2stop**-Befehle fehl.

DB2_DIAGPATH

- Betriebssystem: Alle
- Standardwert: Der Standardwert ist das Verzeichnis db2dump der Instanz unter UNIX- und Linux-Betriebssystemen und das Verzeichnis db2 der Instanz unter Windows-Betriebssystemen.
- Dieser Parameter gilt nur für ODBC- und CLI-Anwendungen.

  Mit diesem Parameter können Sie einen vollständig qualifizierten Pfad für DB2-Diagnosedaten angeben. Im angegebenen Verzeichnis können abhängig von Ihrer Plattform Speicherauszugsdateien, Trapdateien, eine Fehlerprotokolldatei, eine Benachrichtigungsdatei und eine Alertprotokolldatei gespeichert werden.

Die Einstellung dieser Umgebungsvariablen hat im Rahmen der jeweiligen Umgebung für ODBC- und CLI-Anwendungen die gleiche Wirkung wie die Einstellung des Konfigurationsparameters **diagpath** des DB2-Datenbankmanagers und wie die Einstellung des CLI/ODBC-Konfigurationsschlüsselworts **DiagPath**.

DB2DOMAINLIST

- Betriebssystem: Alle
- Standardwert: NULL, Werte: Eine Liste mit Windows-Domänennamen, die durch Kommas („,") getrennt werden.
- Diese Variable definiert eine oder mehrere Windows-Domänen. Die Liste, die auf dem Server verwaltet wird, definiert die Domänen, in denen die anfordernde Benutzer-ID authentifiziert wird. Nur CONNECT- oder ATTACH-Verbindungsanforderungen von Benutzern, die zu diesen Domänen gehören, werden akzeptiert.

  Diese Variable ist nur wirksam, wenn in der Datenbankmanagerkonfiguration der Authentifizierungstyp CLIENT definiert ist. Sie wird benötigt, wenn die Funktion zur einmaligen Anmeldung (Single Sign-on) über einen Windows-Desktop in einer Windows-Domänenumgebung erforderlich ist.

  DB2DOMAINLIST wird unterstützt, wenn entweder der Client oder der Server in einer Windows-Umgebung ausgeführt wird.

DB2ENVLIST

- Betriebssystem: UNIX
- Standardwert: NULL
- Diese Variable listet spezifische Variablennamen für gespeicherte Prozeduren oder für benutzerdefinierte Funktionen auf. Standardmäßig filtert der Befehl **db2start** alle Benutzerumgebungsvariablen außer denjenigen heraus, die das Präfix „DB2" oder „db2" haben. Wenn bestimmte Umgebungsvariablen entweder an gespeicherte Prozeduren oder an benutzerdefinierte Funktionen übergeben werden müssen, können Sie die Variablennamen in der Umgebungsvariablen **DB2ENVLIST** auflisten. Trennen Sie die einzelnen Variablennamen durch ein oder mehrere Leerzeichen.

DB2INSTANCE

- Betriebssystem: Alle
- Standardwert: **DB2INSTDEF** unter 32-Bit-Windows-Betriebssystemen
- Diese Umgebungsvariable gibt die Instanz an, die standardmäßig aktiv ist. Unter UNIX müssen Benutzer einen Wert für **DB2INSTANCE** angeben.

  Anmerkung: Sie können den Befehl **db2set** zum Aktualisieren dieser Registrierdatenbankvariablen verwenden. Weitere Informationen finden Sie in „Ermitteln der aktuellen Instanz" auf Seite 624 und „Definieren von Umgebungsvariablen außerhalb der Profilregistrierdatenbanken unter Windows" auf Seite 622.

DB2INSTPROF

- Betriebssystem: Windows
- Standardwert: Dokumente und Einstellungen\All Users\Anwendungsdaten\IBM\DB2*Name der Kopie* (Windows XP, Windows 2003), ProgramData\IBM\DB2*Name der Kopie* (Windows Vista)

- Diese Umgebungsvariable gibt die Position des Instanzverzeichnisses unter Windows-Betriebssystemen an. Das Instanzverzeichnis kann sich nicht unter dem Verzeichnis `sqllib` befinden (dies gilt auch für andere Benutzerdatendateien).

DB2LDAPSecurityConfig

- Betriebssystem: Alle
- Standardwert: `NULL`, Werte: gültiger Name und Pfad der Konfigurationsdatei für IBM LDAP-Sicherheits-Plug-ins
- Diese Variable dient zur Angabe der Position der Konfigurationsdatei für IBM LDAP-Sicherheits-Plug-ins. Wenn diese Variable nicht definiert ist, hat die Konfigurationsdatei für IBM LDAP-Sicherheits-Plug-ins den Namen `IBMLDAPSecurity.ini` und befindet sich an einer der folgenden Positionen:
  - Unter Linux- und UNIX-Betriebssystemen: *INSTHOME*`/sqllib/cfg/`
  - Unter Windows-Betriebssystemen: `%`**DB2PATH**`%\cfg\`

  Unter Windows-Betriebssystemen sollte diese Variable in der globalen Systemumgebung definiert werden, um sicherzustellen, dass sie vom DB2-Service berücksichtigt wird.

DB2LIBPATH

- Betriebssystem: UNIX
- Standardwert: `NULL`
- DB2 bildet einen eigenen gemeinsamen Bibliothekspfad. Wenn Sie dem Bibliothekspfad der Steuerkomponente einen Pfad (PATH) hinzufügen möchten (z. B. erfordert eine benutzerdefinierte Funktion unter AIX einen bestimmten Eintrag in **LIBPATH**), müssen Sie die Variable **DB2LIBPATH** definieren. Der tatsächliche Wert der Variablen **DB2LIBPATH** wird an das Ende des von DB2 gebildeten gemeinsam genutzten Bibliothekspfad angehängt.

DB2LOGINRESTRICTIONS

- Betriebssystem: AIX
- Standardwert: `LOCAL`, Werte: `LOCAL`, `REMOTE`, `SU`, `NONE`
- Durch diese Registrierdatenbankvariable können Sie eine AIX-Betriebssystems-API namens loginrestrictions() verwenden. Diese API bestimmt, ob ein Benutzer für den Zugriff auf das System berechtigt ist. Durch Aufrufen dieser API kann die DB2-Datenbanksicherheitsfunktion die Anmeldeeinschränkungen umsetzen, die durch das Betriebssystem definiert sind. Es gibt unterschiedliche Werte, die an diese API übergeben werden können, wenn diese Registrierdatenbankvariable verwendet wird. Diese Werte sind:
  - `REMOTE`

    DB2 setzt Anmeldeeinschränkungen nur um, um zu überprüfen, ob das Konto für ferne Anmeldevorgänge über die Programme **rlogind** bzw. **telnetd** verwendet werden kann.
  - `SU`

    DB2 Version 9.1 setzt **su**-Einschränkungen nur um, um zu überprüfen, ob der Befehl **su** zulässig ist und ob der aktuelle Prozess über eine Gruppen-ID verfügt, die den Befehl **su** aufrufen kann, um zu dem Konto umzuschalten.
  - `NONE`

    DB2 setzt keinerlei Anmeldeeinschränkungen um.

– LOCAL (oder die Variable ist nicht definiert)

DB2 setzt Anmeldeeinschränkungen nur um, um zu überprüfen, ob für dieses Konto lokale Anmeldungen erlaubt sind. Dies ist das normale Verhalten bei einer Anmeldung.

Unabhängig davon, welche dieser Optionen Sie definieren, können Benutzerkonten bzw. -IDs, die über die angegebenen Zugriffsrechte verfügen, DB2 sowohl lokal auf dem Server als auch über ferne Clients erfolgreich verwenden. Eine Beschreibung der API loginrestrictions() finden Sie in der AIX-Dokumentation.

DB2NODE

- Betriebssystem: Alle
- Standardwert: NULL, Werte: 1 bis 999
- Dient zur Angabe des logischen Zielknotens eines Datenbankpartitionsservers, zu dem eine Verbindung über CONNECT oder ATTACH hergestellt werden soll. Wenn diese Variable nicht definiert wird, wird als logischer Zielknoten standardmäßig der logische Knoten angenommen, der auf der Maschine mit Port 0 definiert ist. In einer Umgebung mit partitionierten Datenbanken können sich die Verbindungseinstellungen auf die Herstellung gesicherter Verbindungen auswirken. Wenn die Variable **DB2NODE** zum Beispiel auf einen Knoten gesetzt wird, sodass die Herstellung einer Verbindung auf diesem Knoten den Weg über einen Zwischenknoten (Hop-Knoten) erfordert, wird die IP-Adresse dieses Zwischenknotens und das Kommunikationsprotokoll, das zur Kommunikation zwischen dem Zwischenknoten und dem Verbindungsknoten verwendet wird, bei der Bewertung dieser Verbindung geprüft, um festzustellen, ob sie als gesicherte Verbindung markiert werden kann oder nicht. Das bedeutet, dass nicht der ursprüngliche Knoten betrachtet wird, von dem aus die Verbindung eingeleitet wurde, sondern der Zwischenknoten.

  Anmerkung: Sie können den Befehl **db2set** zum Aktualisieren dieser Registrierdatenbankvariablen verwenden. Weitere Informationen hierzu finden Sie in „Definieren von Umgebungsvariablen außerhalb der Profilregistrierdatenbanken unter Windows” auf Seite 622.

DB2OPTIONS

- Betriebssystem: Alle
- Standardwert: NULL
- Dient zum Festlegen der Optionen für den Befehlszeilenprozessor.

DB2_PARALLEL_IO

- Betriebssystem: Alle
- Standardwert: NULL oder * (in einer DB2 pureScale-Umgebung) Werte: *TablespaceID*:[*n*],... – eine durch Kommata getrennte Liste definierter Tabellenbereiche (angegeben durch ihre numerische Tabellenbereichs-ID). Wenn PREFETCHSIZE eines Tabellenbereichs auf AUTOMATIC gesetzt ist, können Sie dem DB2-Datenbankmanager die Anzahl Platten pro Container für diesen Tabellenbereich mitteilen, indem Sie die Tabellenbereichs-ID, gefolgt von einem Doppelpunkt, gefolgt von der Anzahl Platten pro Container *n* angeben. Wird *n* nicht angegeben, ist der Standardwert 6.

  Sie können *tabellenbereichs-id* durch einen Stern (*) ersetzen, um alle Tabellenbereiche anzugeben. Wenn Sie zum Beispiel **DB2_PARALLEL_IO** =* angeben, verwenden alle Tabellenbereiche 6 als Anzahl von Platten pro

Container. Wenn Sie sowohl einen Stern als auch eine Tabellenbereichs-ID angeben, hat die Einstellung der Tabellenbereichs-ID Vorrang. Beispiel: Bei der Angabe **DB2_PARALLEL_IO** =*,1:3 verwenden alle Tabellenbereiche 6 als Anzahl der Platten pro Container mit Ausnahme des Tabellenbereichs 1, der 3 Platten pro Container verwendet.

- Diese Registrierdatenbankvariable dient zur Änderung der Art und Weise, wie DB2 die E/A-Parallelität eines Tabellenbereichs berechnet. Wenn die E/A-Parallelität aktiviert ist (entweder implizit durch die Verwendung mehrerer Container oder explizit durch die Einstellung der Variablen **DB2_PARALLEL_IO**), wird sie umgesetzt, indem die richtige Anzahl von Vorablesezugriffsanforderungen abgesetzt wird. Jede Vorablesezugriffsanforderung ist eine Anforderung für einen EXTENTSIZE großen Speicherbereich von Seiten. Betrachten Sie zum Beispiel den Fall, dass ein Tabellenbereich zwei Container hat und der Wert für PREFETCHSIZE das Vierfache des Werts für EXTENTSIZE beträgt. Wenn die Registrierdatenbankvariable definiert ist, wird eine Vorableseanforderung für diesen Tabellenbereich in vier Anforderungen zerlegt (ein EXTENTSIZE großer Speicherbereich pro Anforderung). Dies bietet die Möglichkeit, dass vier Vorablesefunktionen die Anforderungen parallel bedienen.

  Es kann sinnvoll sein, die Registrierdatenbankvariable zu definieren, wenn die einzelnen Container in dem Tabellenbereich über mehrere physische Datenträger einheitenübergreifend verteilt gespeichert werden (Stripe-Set) oder wenn der Container in einem Tabellenbereich auf einer einzelnen RAID-Einheit erstellt wird, die aus mehr als einer physischen Platte besteht.

  Wenn diese Registrierdatenbankvariable nicht definiert ist, entspricht der Grad der Parallelität eines Tabellenbereichs der Anzahl der Container des Tabellenbereichs. Wenn **DB2_PARALLEL_IO** zum Beispiel auf den Wert NULL gesetzt wird und ein Tabellenbereich über vier Container verfügt, werden vier Vorablesezugriffsanforderungen in der Größe von EXTENTSIZE abgesetzt. Oder wenn ein Tabellenbereich zwei Container hat und der Wert für PREFETCHSIZE das Vierfache des Werts für EXTENTSIZE beträgt, wird eine Vorableseanforderung für diesen Tabellenbereich in zwei Anforderungen zerlegt (jeweils eine Anforderung für zwei EXTENTSIZE große Speicherbereiche).

  Wenn diese Registrierdatenbankvariable definiert ist und PREFETCHSIZE für die Tabelle nicht auf AUTOMATIC gesetzt ist, ist der Grad der Parallelität des Tabellenbereichs gleich dem Wert für PREFETCHSIZE dividiert durch den Wert von EXTENTSIZE. Wenn **DB2_PARALLEL_IO** zum Beispiel für einen Tabellenbereich definiert ist, dessen PREFETCHSIZE-Wert 160 und dessen EXTENTSIZE-Wert 32 Seiten beträgt, werden fünf EXTENTSIZE große Vorablesezugriffsanforderungen abgesetzt.

  Wenn diese Registrierdatenbankvariable definiert ist und PREFETCHSIZE für den Tabellenbereich auf den Wert AUTOMATIC gesetzt ist, berechnet DB2 die Größe des Vorablesezugriffs des Tabellenbereichs automatisch. In der folgenden Tabelle sind die verschiedenen verfügbaren Optionen sowie die Berechnung der Parallelität für die einzelnen Fälle zusammengefasst:

Tabelle 123. Berechnung der Parallelität

PREFETCHSIZE des Tabellenbereichs	**DB2_PARALLEL_IO-Einstellung**	**Die Parallelität ist gleich:**
AUTOMATIC	Nicht definiert	Anzahl Container
AUTOMATIC	*Tabellenbereichs-ID*	Anzahl Container * 6

Tabelle 123. Berechnung der Parallelität (Forts.)

PREFETCHSIZE des Tabellenbereichs	**DB2_PARALLEL_IO-Einstellung**	**Die Parallelität ist gleich:**
AUTOMATIC	*Tabellenbereichs-ID*:*n*	Anzahl Container * *n*
Nicht AUTOMATIC	Nicht definiert	Anzahl Container
Nicht AUTOMATIC	*Tabellenbereichs-ID*	PREFETCHSIZE / EXTENTSIZE
Nicht AUTOMATIC	*Tabellenbereichs-ID*:*n*	PREFETCHSIZE / EXTENTSIZE

In einigen Szenarios kann die Verwendung dieser Variablen zu einer Konkurrenzsituation für Datenträger führen. Wenn z. B. ein Tabellenbereich zwei Container hat und für jeden der beiden Container jeweils ein einzelner dedizierter Datenträger vorhanden wäre, kann eine Definition der Registrierdatenbankvariablen zu Konkurrenzsituationen beim Zugriff auf diese Datenträger führen, da jeweils zwei Vorablesefunktionen versuchen würden, gleichzeitig auf jeden der beiden Datenträger zuzugreifen. Wenn hingegen jeder der beiden Container über mehrere Datenträger einheitenübergreifend gespeichert wäre, würde eine Definition der Registrierdatenbankvariablen potenziell einen gleichzeitigen Zugriff auf vier verschiedene Datenträger ermöglichen.

Zur Aktivierung von Änderungen an dieser Registrierdatenbankvariablen führen Sie den Befehl **db2stop** und anschließend den Befehl **db2start** aus.

DB2PATH

- Betriebssystem: Windows
- Standardwert: je nach Betriebssystem unterschiedlich
- Diese Umgebungsvariable wird zur Angabe des Verzeichnisses verwendet, in dem das Produkt auf 32-Bit-Windows-Betriebssystemen installiert ist.

DB2_PMAP_COMPATIBILITY

- Betriebssystem: Alle
- Standardwert: `ON`, Werte: `ON` oder `OFF`
- Diese Variable ermöglicht es Benutzern, die APIs sqlugtpi und sqlugrpn weiterhin zu verwenden, sodass die Verteilungsdaten für Tabellen und die Datenbankpartitionsnummer und die Nummer des Datenbankpartitionsservers für Zeilen zurückgegeben werden. Die Standardeinstellung `ON` gibt an, dass der Umfang der Verteilungszuordnung bei 4 096 Einträgen bleibt (wie bei den Versionen vor Version 9.7). Wird diese Variable mit `OFF` definiert, wird der Umfang der Verteilungszuordnung für neue oder aktualisierte Datenbanken auf 32 768 Einträge erhöht (kennzeichnend für Version 9.7). Diese umfangreiche Verteilungszuordnung setzt voraus, dass Sie die neuen APIs db2GetDistMap und db2GetRowPartNum und verwenden.
- Änderungen an dieser Variable werden sofort wirksam und gelten für alle zukünftigen kompilierten SQL-Anweisungen. Es ist nicht nötig, die Instanz erneut zu starten oder den Befehl **db2set** mit dem Parameter **-immediate** abzusetzen.

DB2PROCESSORS

- Betriebssystem: Windows

- Standardwert: NULL, Werte: 0–*n*-1 (mit *n* = Anzahl Prozessoren)
- Diese Variable definiert die Prozessaffinitätsmaske für einen bestimmten Prozess **db2syscs**. In Umgebungen mit mehreren logischen Knoten wird diese Variable verwendet, um einem Prozessor bzw. einer Gruppe von Prozessoren einen logischen Knoten zuzuordnen.

  Wird diese Variable angegeben, ruft DB2 die API SetProcessAffinity-Mask() auf. Wird diese Variable nicht angegeben, wird der Prozess **db2syscs** allen Prozessoren auf dem Server zugeordnet.

DB2RCMD_LEGACY_MODE

- Betriebssystem: Windows
- Standardwert: NULL, Werte: YES, ON, TRUE bzw. 1 oder NO, OFF, FALSE bzw. 0
- Diese Variable gibt Benutzern die Möglichkeit, die erweiterten Sicherheitsmerkmale für den DB2 Remote Command Service zu aktivieren oder zu inaktivieren. Zur Ausführung des DB2 Remote Command Service in einem sicheren Modus setzen Sie die Variable **DB2RCMD_LEGACY_MODE** auf NO, OFF, FALSE, 0 oder NULL. Zur Ausführung im traditionellen Modus (ohne erweiterte Sicherheit) setzen Sie die Variable **DB2RCMD_LEGACY_MODE** auf YES, ON, TRUE oder 1. Der sichere Modus steht nur zur Verfügung, wenn Ihr Domänencontroller mit Windows 2000 oder einer späteren Version des Betriebssystems ausgeführt wird.

  Anmerkung: Wenn die Variable **DB2RCMD_LEGACY_MODE** auf YES, ON, TRUE oder 1 gesetzt ist, werden alle Anforderungen, die an den DB2 Remote Command Service gesendet werden, unter dem Kontext des Anforderers verarbeitet. Zu diesem Zweck müssen Sie zulassen, dass die Maschine und/oder das Serviceanmeldekonto die Identität des Clients annehmen, indem Sie das Maschinenkonto und das Serviceanmeldekonto auf dem Domänencontroller aktivieren.

  Anmerkung: Wenn die Variable **DB2RCMD_LEGACY_MODE** auf NO, OFF, FALSE oder 0 gesetzt ist, müssen Sie über die Berechtigung SYSADM verfügen, damit der DB2 Remote Command Service Befehle für Sie ausführen kann.

DB2RESILIENCE

- Betriebssystem: Alle
- Standardwert: ON, Werte: ON (TRUE bzw. 1) oder OFF (FALSE bzw. 0)
- Mithilfe dieser Registrierdatenbankvariablen kann gesteuert werden, ob physische Lesefehler toleriert werden. In diesem Fall wird die erweiterte Trap-Recovery aktiviert. Das Standardverhalten besteht darin, Lesefehler zu tolerieren und die erweiterte Trap-Recovery zu aktivieren. Wenn Sie zum Verhalten früherer Releases zurückkehren und den Datenbankmanager veranlassen wollen, die Instanz herunterzufahren, setzen Sie die Registrierdatenbankvariable auf den Wert OFF. Diese Registrierdatenbankvariable hat keine Auswirkung auf die vorhandene Speicherschlüsselunterstützung.

DB2_RESTORE_GRANT_ADMIN_AUTHORITIES

- Betriebssystem: Alle
- Standardwert: OFF, Werte: ON oder OFF
- Wenn für **DB2_RESTORE_GRANT_ADMIN_AUTHORITIES** der Wert ON angegeben wird und Sie eine Wiederherstellung (Restore) in einer neuen oder vor-

handenen Datenbank durchführen, werden Ihnen die Berechtigungen SECADM, DBADM, DATAACCESS und ACCESSCTRL erteilt.

- Die folgenden Restoremethoden werden unterstützt, wenn für **DB2_RESTORE_GRANT_ADMIN_AUTHORITIES** der Wert ON angegeben ist:
  - Backups von geteilten Spiegeldatenbanken
  - ACS-Momentaufnahmebackups
  - Online- und Offline-Backups von Datenbanken mit dem Befehl **RESTORE DATABASE**

  Anmerkung: Beachten Sie, dass diese Variable keine Auswirkung auf Tabellenbereichsrestores hat; dem Benutzer, der die Restoreoperation ausführt, werden keine zusätzlichen Berechtigungen erteilt.
- Wenn für **DB2_WORKLOAD** der Wert SAP definiert wird, wird **DB2_RESTORE_GRANT_ADMIN_AUTHORITIES** auf den Wert ON gesetzt.

DB2SYSTEM

- Betriebssystem: Windows und UNIX
- Standardwert: NULL
- Gibt den Namen an, der von Ihren Benutzern und Datenbankadministratoren zur Identifizierung des DB2-Datenbankserversystems verwendet wird. Dieser Name sollte nach Möglichkeit innerhalb Ihres Netzes eindeutig sein.

  Dieser Name unterstützt Benutzer bei der Identifizierung des Systems, das die Datenbank enthält, auf die Sie zugreifen wollen. Bei der Installation wird für **DB2SYSTEM** wie folgt ein Wert festgelegt:
  - Unter Windows setzt das Installationsprogramm diesen Parameter auf den Wert des Computernamens, der für das Windows-System angegeben ist.
  - Auf UNIX-Systemen wird für diesen Parameter der TCP/IP-Hostname des UNIX-Systems definiert.

DB2_UPDDBCFG_SINGLE_DBPARTITION

- Betriebssystem: Alle
- Standardwert: nicht definiert, Werte: 0/FALSE/NO, 1/TRUE/YES
- Mithilfe der Variablen **DB2_UPDDBCFG_SINGLE_DBPARTITION** können Sie zum Verhalten früherer Versionen von DB2 zurückkehren, bei dem Aktualisierungen an einer Datenbankkonfiguration nur für die lokale Datenbankpartition bzw. für die durch die Registrierdatenbankvariable **DB2NODE** angegebene Datenbankpartition gelten. Dies ermöglicht Abwärtskompatibilität zur Unterstützung vorhandener Befehlsscripts oder Anwendungen, die dieses Verhalten voraussetzen.

  Durch Setzen dieser Registrierdatenbankvariablen auf 1, TRUE oder YES können Sie angeben, dass alle Aktualisierungen und Zurücksetzungen an Ihrer Datenbank nur eine bestimmte Partition betreffen. Wenn die Variable nicht festgelegt wird (dies ist der Standardwert), betreffen Aktualisierungen bzw. Änderungen an einer Datenbankkonfiguration alle Datenbankpartitionen, wenn Sie keine Partitionsklausel angeben.

Anmerkung: Diese Variable bezieht sich nicht auf Aktualisierungs- oder Rücksetzanforderungen, die durch Aufrufen von ADMIN_CMD-Routinen erfolgen.

DB2_USE_PAGE_CONTAINER_TAG

- Betriebssystem: Alle
- Standardwert: NULL, Werte: ON, NULL
- Standardmäßig speichert DB2 eine Containerkennung im ersten Speicherbereich jedes DMS-Containers. Dabei spielt es keine Rolle, ob es sich um eine Datei oder eine Einheit handelt. Die Containerkennung bildet die Metadaten für den Container. Vor DB2 Version 8.1 wurde die Containerkennung auf einer einzigen Seite gespeichert und erforderte so weniger Speicherplatz im Container. Wenn die Containerkennung weiterhin in einer einzigen Seite gespeichert werden soll, setzen Sie die Variable **DB2_USE_PAGE_CONTAINER_TAG** auf den Wert ON.

  Wenn Sie jedoch diese Registrierdatenbankvariable auf ON setzen, wenn Sie RAID-Einheiten als Container verwenden, kann sich die E/A-Leistung verschlechtern. Da Sie für RAID-Einheiten Tabellenbereiche mit einem EXTENTSIZE-Wert erstellen, der der Stripegröße oder einem Vielfachen der Stripegröße der RAID-Einheiten entspricht, führt die Einstellung der Variablen **DB2_USE_PAGE_CONTAINER_TAG** auf ON dazu, dass sich die EXTENTSIZE großen Speicherbereiche nicht an den RAID-Stripes (einheitenübergreifend gespeicherten Datenblöcken) ausrichten. Infolgedessen muss eine E/A-Anforderung eventuell auf mehr physische Platten zugreifen, als es optimal der Fall wäre. Benutzern wird ausdrücklich davon abgeraten, diese Registrierdatenbankvariable zu aktivieren, sofern Sie nicht unter sehr eingeschränkten Speicherverhältnissen arbeiten oder eine Funktionsweise benötigen, die mit der früherer Versionen (vor Version 8) übereinstimmt.

  Zur Aktivierung von Änderungen an dieser Registrierdatenbankvariablen führen Sie den Befehl **db2stop** und anschließend den Befehl **db2start** aus.

DB2_WORKLOAD

- Betriebssystem: Alle
- Standardwert: nicht definiert, Werte: 1C, CM, COGNOS_CS, FILENET_CM, INFOR_ERP_LN, MAXIMO, MDM, SAP, TPM, WAS, WC oder WP
- Jeder Wert für die Variable **DB2_WORKLOAD** stellt eine bestimmte Gruppierung verschiedener Registrierdatenbankvariablen mit vordefinierten Einstellungen dar.
- Die folgenden Werte sind gültig:

  1C Verwenden Sie diese Einstellung, wenn Sie eine Gruppe von Registrierdatenbankvariablen in Ihrer Datenbank für 1C-Anwendungen konfigurieren wollen.

  CM Verwenden Sie diese Einstellung, wenn Sie eine Gruppe von Registrierdatenbankvariablen in Ihrer Datenbank für IBM Content Manager konfigurieren wollen.

  COGNOS_CS
  Verwenden Sie diese Einstellung, wenn Sie eine Gruppe von Registrierdatenbankvariablen in Ihrer Datenbank für Cognos Content Server konfigurieren wollen.

  FILENET_CM
  Verwenden Sie diese Einstellung, wenn Sie eine Gruppe von Registrierdatenbankvariablen in Ihrer Datenbank für FileNet Content Manager konfigurieren wollen.

INFOR_ERP_LN
Verwenden Sie diese Einstellung, wenn Sie eine Gruppe von Registrierdatenbankvariablen in Ihrer Datenbank für Infor ERP Baan konfigurieren wollen.

MAXIMO
Verwenden Sie diese Einstellung, wenn Sie eine Gruppe von Registrierdatenbankvariablen in Ihrer Datenbank für Maximo konfigurieren wollen.

MDM Verwenden Sie diese Einstellung, wenn Sie eine Gruppe von Registrierdatenbankvariablen in Ihrer Datenbank für Master Data Management konfigurieren wollen.

SAP Verwenden Sie diese Einstellung, wenn Sie eine Gruppe von Registrierdatenbankvariablen in Ihrer Datenbank für die SAP-Umgebung konfigurieren wollen.

Wenn Sie **DB2_WORKLOAD**=SAP festgelegt haben, werden der Benutzertabellenbereich SYSTOOLSPACE und der Tabellenbereich für temporäre Benutzertabellen SYSTOOLSTMPSPACE nicht automatisch erstellt. Diese Tabellenbereiche werden für Tabellen verwendet, die von den folgenden Assistenten, Dienstprogrammen und Funktionen automatisch erstellt werden:

- Automatische Verwaltung
- Gespeicherte Prozedur SYSINSTALLOBJECTS, wenn der Eingabeparameter für Tabellenbereich nicht angegeben wird
- Gespeicherte Prozedur GET_DBSIZE_INFO

Ohne die Tabellenbereiche SYSTOOLSPACE und SYSTOOLSTMPSPACE können Sie diese Assistenten, Dienstprogramme und Funktionen nicht nutzen.

Wenn Sie diese Assistenten, Dienstprogramme oder Funktionen verwenden möchten, führen Sie eine der folgenden Aktionen aus:

- Erstellen Sie den Tabellenbereich SYSTOOLSPACE manuell, sodass er die Objekte aufnehmen kann, die von den Tools benötigt werden. (In einer Umgebung mit partitionierten Datenbanken erstellen Sie diesen Tabellenbereich in der Katalogpartition.) Beispiel:

```
CREATE REGULAR TABLESPACE SYSTOOLSPACE
IN IBMCATGROUP
MANAGED BY SYSTEM
USING ('SYSTOOLSPACE')
```

- Rufen Sie unter Angabe eines gültigen Tabellenbereichs die gespeicherte Prozedur SYSINSTALLOBJECTS auf, um die Objekte für die Tools zu erstellen, und geben Sie den Bezeichner für das bestimmte Tool an. SYSINSTALLOBJECTS erstellt einen Tabellenbereich für Sie. Wenn Sie den Tabellenbereich SYSTOOLSSPACE nicht für die Objekte verwenden wollen, geben Sie einen anderen benutzerdefinierten Tabellenbereich an.

Nachdem Sie mindestens eine dieser Optionen ausgeführt haben, erstellen Sie den Tabellenbereich SYSTOOLSTMPSPACE für temporäre Tabellen (ebenfalls in der Katalogpartition, wenn Sie in einer Umgebung mit partitionierten Datenbanken arbeiten). Beispiel:

```
CREATE USER TEMPORARY TABLESPACE SYSTOOLSTMPSPACE
IN IBMCATGROUP
MANAGED BY SYSTEM
USING ('SYSTOOLSTMPSPACE')
```

Wenn der Tabellenbereich SYSTOOLSPACE und der temporäre Tabellenbereich SYSTOOLSTMPSPACE erstellt sind, können Sie die oben genannten Assistenten, Dienstprogramme bzw. Funktionen verwenden.

TPM Verwenden Sie diese Einstellung, wenn Sie eine Gruppe von Registrierdatenbankvariablen in Ihrer Datenbank für Tivoli Provisioning Manager konfigurieren wollen.

WAS Verwenden Sie diese Einstellung, wenn Sie eine Gruppe von Registrierdatenbankvariablen in Ihrer Datenbank für WebSphere Application Server konfigurieren wollen.

WC Verwenden Sie diese Einstellung, wenn Sie eine Gruppe von Registrierdatenbankvariablen in Ihrer Datenbank für WebSphere Commerce konfigurieren wollen.

WP Verwenden Sie diese Einstellung, wenn Sie eine Gruppe von Registrierdatenbankvariablen in Ihrer Datenbank für WebSphere Portal konfigurieren wollen.

Kommunikationsvariablen

DB2CHECKCLIENTINTERVAL

- Betriebssystem: Alle, nur Server
- Standardwert: 100, Werte: Ein numerischer Wert größer-gleich null
- Diese Variable gibt die Häufigkeit von TCP/IP-Clientverbindungsprüfungen während einer aktiven Transaktion an. Sie ermöglicht das frühe Erkennen einer Clientbeendigung, ohne den Abschluss der Abfrage abzuwarten. Wird diese Variable auf `0` gesetzt, wird keine Prüfung durchgeführt.

  Niedrigere Werte bedeuten häufigere Überprüfungen. Verwenden Sie `100` als Richtwert für geringe Häufigkeit, `50` für mittlere Häufigkeit und `10` für hohe Häufigkeit. Der Wert wird in einer internen DB2-Metrik gemessen. Die Werte stellen eine lineare Skala dar. Dies bedeutet, dass eine Erhöhung des Werts von `50` auf `100` die Länge des Intervalls verdoppelt. Je häufiger Sie den Clientstatus während der Ausführung einer Datenbankanforderung prüfen, desto länger dauert die Ausführung von Abfragen. Wenn die DB2-Auslastung hoch ist (d. h., wenn viele interne Anforderungen verarbeitet werden), hat die Einstellung von **DB2CHECKCLIENTINTERVAL** auf einen niedrigen Wert eine größere Auswirkung auf die Leistung als in einer Situation, in der die Auslastung gering ist.

DB2COMM

- Betriebssystem: Alle, nur Server
- Standardwert: NULL, Werte: `NPIPE`, `TCPIP`, `SSL`
- Diese Variable gibt die Kommunikationsmanager an, die gestartet werden, wenn der Datenbankmanager gestartet wird. Wenn diese Variable nicht definiert wird, werden auf dem Server keine DB2-Kommunikationsmanager gestartet.

DB2FCMCOMM

- Betriebssystem: Alle unterstützten DB2 Enterprise Server Edition-Plattformen
- Standardwert: TCPIP4, Werte: TCPIP4 oder TCPIP6
- Diese Variable gibt an, wie die Hostnamen in der Datei db2nodes.cfg aufgelöst werden sollen. Alle Hostnamen werden auf der Basis von IPv4 oder IPv6 aufgelöst. Wenn in der Datei db2nodes.cfg anstelle eines Hostnamens eine IP-Adresse angegeben wird, dann bestimmt das Format dieser IP-Adresse, ob IPv4 oder IPv6 verwendet wird. Wenn **DB2FCMCOMM** nicht definiert ist, dann können entsprechend der Standardeinstellung IPv4 nur IPv4-Hosts gestartet werden.

  Anmerkung: Wenn das IP-Format, das auf der Basis des in db2nodes.cfg angegebenen Hostnamens aufgelöst wurde, oder das direkt in db2nodes.cfg angegebene IP-Format nicht mit der Einstellung von **DB2FCMCOMM** übereinstimmt, schlägt die Ausführung von **db2start** fehl.

DB2_FORCE_NLS_CACHE

- Betriebssystem: AIX, HP_UX, Solaris
- Standardwert: FALSE, Werte: TRUE oder FALSE
- Diese Variable dient zur Verhinderung der Möglichkeit von Sperrkonflikten in Multithread-Anwendungen. Wenn diese Registrierdatenbankvariable den Wert TRUE hat, werden die Informationen zur Codepage und zum Gebietscode beim Erstzugriff eines Threads auf die Informationen im Cache gespeichert. Von da an werden die im Cache gespeicherten Informationen für jeden anderen Thread verwendet, der diese Informationen anfordert. Dadurch wird der Sperrkonflikt beseitigt, was in bestimmten Situationen zu einem Leistungsvorteil führt. Diese Einstellung sollte nicht verwendet werden, wenn die Anwendung länderspezifische Angaben (Locale-Einstellungen) zwischen Verbindungen ändert. In einer solchen Situation ist sie wahrscheinlich kaum erforderlich, da Multithread-Anwendungen ihre länderspezifischen Einstellungen in der Regel nicht ändern, weil dies nicht *threadsicher* ist.

DB2_PMODEL_SETTINGS

- Betriebssystem: Alle
- Änderungen an dieser Variable werden sofort wirksam und gelten für alle zukünftigen kompilierten SQL-Anweisungen. Es ist nicht nötig, die Instanz erneut zu starten oder den Befehl **db2set** mit dem Parameter **-immediate** abzusetzen.
- Diese Registrierdatenbankvariable steuert eine Gruppe von Parametern, mit denen Sie das Verhalten verschiedener Aspekte der internen DB2-Infrastruktur ändern können. Trennen Sie die Parameter wie im folgenden Beispiel durch ein Semikolon:

  ```
  db2set
  DB2_PMODEL_SETTINGS=MLN_REMOTE_LISTENER:TRUE;ENHANCED_ROLLBACK:
  TRUE;SRVLST_EQUAL_WEIGHT:TRUE
  ```

- Parameter:

  ENHANCED_ROLLBACK

  – Standardwert: FALSE

  – Werte: TRUE, FALSE

  – Verwenden Sie den Parameter **ENHANCED_ROLLBACK**, um das Rollback-Verhalten für UOWs auf DB2-Servern in Umgebungen mit partitionierten Datenbanken zu verbessern. Wenn Sie diese Option

auf TRUE setzen, werden Rollback-Anforderungen für UOWs nur an logische Datenbankpartitionen gesendet, die an der Transaktion teilgenommen haben.

MLN_REMOTE_LISTENER

- Standardwert: FALSE
- Werte: TRUE, FALSE
- Verwenden Sie den Parameter **MLN_REMOTE_LISTENER**, um einen TCP/IP-Listener auf jeder logischen Datenbankpartition zu starten. Wenn Sie diese Option auf TRUE setzen, können Anwendungen direkt eine Verbindung zu jeder logischen Datenbankpartition herstellen, anstatt Anforderungen über den Datenbankpartitionsserver weiterzuleiten, der dem logischen Port 0 zugeordnet ist.

  Wenn Sie diese Option auf TRUE setzen, stellen Sie sicher, dass die zusätzlichen TCP/IP-Listener keine Ports verwenden, die von anderen Services benötigt werden.

SRVLST_EQUAL_WEIGHT

- Standardwert: FALSE
- Werte: TRUE, FALSE
- Verwenden Sie den Parameter **SRVLST_EQUAL_WEIGHT**, wenn Membergewichtungen ungleich null in der Serverliste immer identisch sein sollen, wodurch das Standardverhalten außer Kraft gesetzt wird, bei dem die Membergewichtungen auf der Basis der Auslastung berechnet werden. Membergewichtungen, wie sie in der Serverliste enthalten sind, werden von einem fernen Client zur Lastverteilung verwendet, wenn die Lastausgleichsfunktion (Workload Balancing, WLB) aktiviert ist.

  Wenn Sie diese Option auf TRUE setzen, sorgt die WLB-Funktion auf dem Client für eine gleichmäßige Lastverteilung unter den Membern - unabhängig von der Memberauslastung.

DB2RSHCMD

- Betriebssystem: UNIX, Linux
- Standardwert: rsh (remsh unter HP-UX), Werte sind vollständige Pfadnamen für rsh, remsh oder ssh
- Standardmäßig verwendet das DB2-Datenbanksystem beim Starten ferner Datenbankpartitionen 'rsh' als Kommunikationsprotokoll. Das Script **db2_all** wird verwendet, um Dienstprogramme und Befehle für alle Datenbankpartitionen auszuführen. Das Einstellen dieser Registrierdatenbankvariablen auf den vollständigen Pfadnamen für ssh veranlasst DB2-Datenbankprodukte zum Beispiel zur Verwendung von ssh als Kommunikationsprotokoll für die angeforderte Ausführung der Dienstprogramme und Befehle. Sie kann auch auf den vollständigen Pfadnamen eines Scripts eingestellt werden, das das ferne Befehlsprogramm mit entsprechenden Standardparametern aufruft. Diese Variable ist nur für partitionierte Datenbanken oder für Umgebungen mit einer einzigen Partition erforderlich, wenn der Befehl **db2start** nicht auf dem Server ausgeführt wird, auf dem das DB2-Produkt installiert ist. Darüber hinaus ist sie für rsh- oder ssh-abhängige Dienstprogramme erforderlich, die eine DB2-Instanz starten, stoppen oder überwachen können, wie zum Beispiel **db2gcf**. Der Instanzeigner muss das angegebene Programm der fernen Shell verwenden können, um sich von jedem DB2-Datenbankknoten aus

bei jedem anderen DB2-Datenbankknoten anzumelden, ohne dass eine weitere Prüfung oder Authentifizierung (d. h. Kennwörter oder Kennwortphrasen) erforderlich wird.

Detaillierte Anweisungen zum Einstellen der Registrierungsdatenbankvariablen **DB2RSHCMD** zur Verwendung einer SSH-Shell mit DB2 finden Sie im White Paper "Configure DB2 Universal Database for UNIX to use OpenSSH" (DB2 Universal Database für UNIX zur Verwendung von OpenSSH konfigurieren).

DB2RSHTIMEOUT

- Betriebssystem: UNIX, Linux
- Standardwert: `30` Sekunden, Werte: `1 - 120`
- Diese Variable ist nur gültig, wenn **DB2RSHCMD** auf einen Wert ungleich NULL gesetzt ist. Diese Registrierdatenbankvariable wird für die Steuerung des Zeitlimitintervalls verwendet, während dessen das DB2-Datenbanksystem auf einen fernen Befehl wartet. Wenn bis zum Ablauf dieses Zeitlimits keine Antwort empfangen wurde, wird angenommen, dass die ferne Datenbankpartition nicht erreichbar und die Operation fehlgeschlagen ist.

  Anmerkung: Bei dem angegebenen Zeitwert handelt es sich nicht um die für die Ausführung des fernen Befehls erforderliche Zeit, sondern um die Zeit, die für die Authentifizierung der Anforderung benötigt wird.

DB2SORCVBUF

- Betriebssystem: Alle
- Standardwert: `65.536`
- Definiert den Wert von TCP/IP-Empfangspuffern.

DB2SOSNDBUF

- Betriebssystem: Alle
- Standardwert: `65.536`
- Definiert den Wert von TCP/IP-Sendepuffern.

DB2TCP_CLIENT_CONTIMEOUT

- Betriebssystem: Alle, nur Client
- Standardwert: `0` (kein Zeitlimit), Werte: `0 - 32.767` Sekunden
- Die Registrierdatenbankvariable **DB2TCP_CLIENT_CONTIMEOUT** gibt die Anzahl Sekunden an, die ein Client auf den Abschluss einer TCP/IP-Verbindungsoperation wartet. Wenn innerhalb der angegebenen Sekunden keine Verbindung hergestellt wird, gibt der DB2-Datenbankmanager den Fehler `-30081 selectForConnectTimeout` zurück.

  Es besteht kein Zeitlimit, wenn die Registrierdatenbankvariable nicht definiert oder auf den Wert `0` gesetzt ist.

  Anmerkung: Auch Betriebssysteme haben einen Wert für ein Verbindungszeitlimit, der früher greifen könnte als das Zeitlimit, das Sie durch **DB2TCP_CLIENT_CONTIMEOUT** definieren. Zum Beispiel hat AIX den Standardwert *tcp_keepinit*=`150` (in Halbsekunden), der den Verbindungsaufbau nach 75 Sekunden beenden würde.

- Änderungen an dieser Variable werden sofort wirksam und gelten für alle zukünftigen kompilierten SQL-Anweisungen. Es ist nicht nötig, die Instanz erneut zu starten oder den Befehl **db2set** mit dem Parameter **-immediate** abzusetzen.

DB2TCP_CLIENT_KEEPALIVE_TIMEOUT

- Betriebssystem: AIX, HP-UX, Linux, Windows (nur Client)
- Standardwert: 15, Werte: 0 - 32.767 Sekunden
- Die Registrierdatenbankvariable **DB2TCP_CLIENT_KEEPALIVE_TIMEOUT** gibt den maximalen Zeitraum in Sekunden an, der vergeht, bis eine nicht antwortende TCP/IP-Clientverbindung oder -anbindung als nicht mehr aktiv erkannt wird. Dies ist das clientseitige Äquivalent zu **DB2TCP_SERVER_KEEPALIVE_TIMEOUT**. Wenn diese Variable nicht festgelegt wird, wird der Standardwert von 15 Sekunden verwendet. Wenn die Variable festgelegt ist, hat diese Variable Vorrang über beliebige 'keepAliveTimeout'-Einstellungen, die in der Datei 'db2dsdriver.cfg' angegeben sind.

  Änderungen an dieser Variable werden sofort wirksam und gelten für alle zukünftigen TCP/IP-Verbindungen und -anbindungen zum bzw. an den Server.

DB2TCP_CLIENT_RCVTIMEOUT

- Betriebssystem: Alle, nur Client
- Standardwert: 0 (kein Zeitlimit), Werte: 0 - 32.767 Sekunden
- Die Registrierdatenbankvariable **DB2TCP_CLIENT_RCVTIMEOUT** gibt die Anzahl Sekunden an, die ein Client auf den Empfang von Daten durch eine TCP/IP-Operation wartet. Wenn innerhalb der angegebenen Sekunden keine Daten vom Server empfangen werden, gibt der DB2-Datenbankmanager den Fehler -30081 selectForRecvTimeout zurück.

  Es besteht kein Zeitlimit, wenn die Registrierdatenbankvariable nicht definiert oder auf den Wert 0 gesetzt ist.

  Anmerkung: Der Wert für **DB2TCP_CLIENT_RCVTIMEOUT** kann durch die CLI überschrieben werden, indem das Schlüsselwort **ReceiveTimeout** für db2cli.ini oder das Verbindungsattribut SQL_ATTR_RECEIVE_TIMEOUT verwendet wird.
- Änderungen an dieser Variable werden sofort wirksam und gelten für alle zukünftigen kompilierten SQL-Anweisungen. Es ist nicht nötig, die Instanz erneut zu starten oder den Befehl **db2set** mit dem Parameter **-immediate** abzusetzen.

DB2TCPCONNMGRS

- Betriebssystem: Alle
- Standardwert: 1 auf seriellen Maschinen; auf symmetrischen Multiprozessormaschinen die aufgerundete Quadratwurzel aus der Anzahl Prozessoren bis zu maximal 16 Verbindungsmanagern. Werte: 1 bis 16
- Wenn die Registrierdatenbankvariable nicht definiert ist, wird die Standardzahl von Verbindungsmanagern erstellt. Wenn die Registrierdatenbankvariable definiert ist, setzt der zugeordnete Wert den Standardwert außer Kraft. Die Zahl der angegebenen TCP/IP-Verbindungsmanager wird bis zu einem Maximum von 16 erstellt. Wenn weniger als 1 angegeben wird, wird **DB2TCPCONNMGRS** auf den Wert 1 gesetzt und eine Warnung protokolliert, dass der Wert außerhalb des gültigen Bereichs liegt. Wenn ein Wert größer als 16 angegeben wird, wird **DB2TCPCONNMGRS** auf

den Wert 16 gesetzt und eine Warnung protokolliert, dass der Wert außerhalb des gültigen Bereichs liegt. Werte zwischen 1 und 16 werden wie angegeben verwendet. Wenn mehr als ein Verbindungsmanager erstellt wird, sollte sich der Verbindungsdurchsatz verbessern, wenn mehrere Clientverbindungen gleichzeitig empfangen werden. Wenn der Benutzer eine SMP-Maschine verwendet oder die Registriervariable **DB2TCPCONNMGRS** geändert hat, können für den TCP/IP-Verbindungsmanager zusätzliche Prozesse (unter UNIX) bzw. Threads (unter Windows-Betriebssystemen) vorhanden sein. Zusätzliche Prozesse oder Threads erfordern zusätzlichen Speicher.

Anmerkung: Wenn die Anzahl der Verbindungsmanager auf den Wert 1 gesetzt wird, kommt es bei Fernverbindungen in Systemen mit zahlreichen Benutzern und/oder häufigem Auf- und Abbau von Verbindungen zu einem Leistungsabfall.

DB2TCP_SERVER_KEEPALIVE_TIMEOUT

- Betriebssystem: AIX, HP-UX, Linux, Windows (nur Server)
- Standardwert: 60, Werte: 0 - 32.767 Sekunden
-

Die Registrierdatenbankvariable **DB2TCP_SERVER_KEEPALIVE_TIMEOUT** gibt den maximalen Zeitraum in Sekunden an, der vergeht, bis eine nicht antwortende TCP/IP-Clientverbindung oder -anbindung als nicht mehr aktiv erkannt wird. Dies ist das clientseitige Äquivalent zu **DB2TCP_CLIENT_KEEPALIVE_TIMEOUT** und 'keepAliveTimeout'. Ist diese Variable nicht festgelegt, wird der Standardwert von 60 Sekunden verwendet.

Änderungen an dieser Variable werden sofort wirksam und gelten für alle zukünftigen TCP/IP-Verbindungen und -anbindungen zum bzw. an den Server. Es ist nicht erforderlich, die Serverinstanz erneut zu starten.

Befehlszeilenvariablen

DB2BQTIME

- Betriebssystem: Alle
- Standardwert=1 Sekunde, Mindestwert: 1 Sekunde
- Diese Variable gibt die Zeitdauer an, die das Front-End des Befehlszeilenprozessors inaktiv bleibt, bevor es prüft, ob der Back-End-Prozess aktiv ist, und eine Verbindung zu diesem Prozess herstellt.

DB2BQTRY

- Betriebssystem: Alle
- Standardwert=60 Wiederholungen, Minimalwert: 0 Wiederholungen
- Diese Variable gibt die Anzahl der Wiederholungen an, mit denen das Front-End des Befehlszeilenprozessors festzustellen versucht, ob der Back-End-Prozess bereits aktiv ist. Diese Variable arbeitet in Verbindung mit der Variablen **DB2BQTIME**.

DB2_CLP_EDITOR

- Betriebssystem: Alle
- Standardwert: Notepad (Windows), vi (UNIX), Werte: Alle gültigen Editoren, die sich im Betriebssystempfad befinden

Anmerkung: Diese Registrierdatenbankvariable wird während der Installation nicht auf den Standardwert gesetzt. Stattdessen verwendet der Code, der diese Variable verwendet, einen Standardwert, wenn die Registrierdatenbankvariable nicht festgelegt ist.

- Diese Variable legt den Editor fest, der bei der Ausführung des Befehls **EDIT** verwendet wird. In einer interaktiven CLP-Sitzung startet der Befehl **EDIT** einen Editor, der mit einem benutzerdefinierten Befehl vorinstalliert ist, der bearbeitet und ausgeführt werden kann.

DB2_CLP_HISTSIZE

- Betriebssystem: Alle
- Standardwert: 20, Werte: 1 bis 500 einschließlich

Anmerkung: Diese Registrierdatenbankvariable wird während der Installation nicht auf den Standardwert gesetzt. Stattdessen verwendet der Code, der diese Variable verwendet, den Standardwert 20, wenn die Registrierdatenbankvariable nicht festgelegt ist oder wenn sie auf einen Wert festgelegt wurde, der sich außerhalb des gültigen Bereichs befindet.

- Diese Variable legt die Anzahl der Befehle fest, die im Befehlsprotokoll während interaktiver CLP-Sitzungen gespeichert werden. Da das Befehlsprotokoll im Speicher gehalten wird, kann ein sehr hoher Wert für diese Variable zu Auswirkungen auf die Leistung führen. Diese hängen von der Anzahl und Länge der Befehle ab, die in einer Sitzung ausgeführt werden.

DB2_CLPPROMPT

- Betriebssystem: Alle
- Standardwert=Keiner (falls kein Standardwert definiert ist, wird „db2 => ” als interaktive Standard-CLP-Eingabeaufforderung verwendet), Werte: Beliebige Textzeichenfolge mit einer Länge von unter 100 Zeichen, die keines oder mehrere der folgenden Token enthält: %i, %d, %ia, %da oder %n. Diese Variable braucht nur dann definiert zu werden, wenn die standardmäßig verwendete interaktive CLP-Eingabeaufforderung (db2 =>) explizit geändert werden soll.
- Mit dieser Registrierdatenbankvariablen können Benutzer die Eingabeaufforderung definieren, die im interaktiven Modus des Befehlszeilenprozessors (CLP) verwendet werden soll. Die Variable kann auf eine beliebige Textzeichenfolge von unter 100 Zeichen gesetzt werden, die keine oder mehrere der optionalen Token %i, %d, %ia, %da oder %n enthält. Im interaktiven CLP-Modus wird die zu verwendende Eingabeaufforderung erstellt, indem die in der Registrierdatenbankvariablen **DB2_CLPPROMPT** angegebene Textzeichenfolge verwendet wird und alle Vorkommen der Token %i, %d, %ia, %da oder %n durch den lokalen Aliasnamen der jeweils verbundenen Instanz, den lokalen Aliasnamen der aktuellen Datenbankverbindung, die Berechtigungs-ID der jeweils verbundenen Instanz, die Berechtigungs-ID der aktuellen Datenbankverbindung bzw. eine neue Zeile (d. h. eine Zeilenschaltung) ersetzt werden.

  Anmerkung:

  1. Wird die Registrierdatenbankvariable **DB2_CLPPROMPT** innerhalb des interaktiven CLP-Modus geändert, tritt der neue Wert für **DB2_CLPPROMPT** erst in Kraft, nachdem der interaktive CLP-Modus geschlossen und erneut geöffnet wurde.

2. Wenn keine Instanzverbindung vorhanden ist, wird %ia durch eine leere Zeichenfolge und %i durch den Wert der Registrierdatenbankvariablen **DB2INSTANCE** ersetzt. Nur auf Windows-Plattformen: Wenn **DB2INSTANCE** nicht definiert ist, wird %i durch den Wert der Registrierdatenbankvariablen **DB2INSTDEF** ersetzt. Ist keine dieser Variablen definiert, wird %i durch eine leere Zeichenfolge ersetzt.
3. Wenn keine Datenbankverbindung vorhanden ist, wird %da durch eine leere Zeichenfolge und %d durch den Wert der Registrierdatenbankvariablen **DB2DBDFT** ersetzt. Ist die Variable **DB2DBDFT** nicht definiert, wird %d durch eine leere Zeichenfolge ersetzt.
4. Die Eingabeaufforderung für interaktive Eingabe zeigt die Werte für die Berechtigungs-IDs, Datenbanknamen und Instanznamen stets in Großbuchstaben an.

DB2IQTIME

- Betriebssystem: Alle
- Standardwert=5 Sekunden, Minimalwert: 1 Sekunde
- Diese Variable definiert die Zeitdauer, die der Back-End-Prozess des Befehlszeilenprozessors an der Eingabewarteschlange darauf wartet, dass der Front-End-Prozess Befehle übergibt.

DB2RQTIME

- Betriebssystem: Alle
- Standardwert=5 Sekunden, Minimalwert: 1 Sekunde
- Diese Variable definiert die Zeitdauer, die der Back-End-Prozess des Befehlszeilenprozessors auf eine Anforderung vom Front-End-Prozess wartet.

Variablen für Umgebungen mit partitionierten Datenbanken

DB2CHGPWD_EEE

- Betriebssystem: DB2 ESE unter AIX, Linux und Windows
- Standardwert=NULL, Werte: YES oder NO
- Diese Variable gibt an, ob Sie zulassen, dass andere Benutzer Kennwörter auf ESE-Systemen unter AIX oder Windows ändern. Es muss sichergestellt werden, dass die Kennwörter für alle Datenbankpartitionen oder Knoten mithilfe eines Windows-Domänencontrollers unter Windows oder LDAP unter AIX zentral verwaltet werden. Wenn Kennwörter nicht zentral verwaltet werden, bleiben sie möglicherweise nicht für alle Datenbankpartitionen bzw. Knoten konsistent. Dies könnte dazu führen, dass ein Kennwort nur in der Datenbankpartition geändert wird, mit der der Benutzer zur Durchführung der Änderung verbunden ist.

DB2_FCM_SETTINGS

- Betriebssystem: Linux
- Standardwert=YES, Werte:
  - FCM_MAXIMIZE_SET_SIZE:[YES|TRUE|NO|FALSE]. Der Standardwert für FCM_MAXIMIZE_SET_SIZE ist YES.
  - FCM_CFG_BASE_AS_FLOOR: [YES|TRUE|NO|FALSE]. Der Standardwert für FCM_CFG_BASE_AS_FLOOR ist NO.
- Sie können die Registrierdatenbankvariable **DB2_FCM_SETTINGS** mit dem Token FCM_MAXIMIZE_SET_SIZE definieren, um einen Standardspeicherbereich von 4 GB für den FCM-Puffer (FCM, Fast Communication Mana-

ger) vorab zuzuordnen. Das Token muss entweder den Wert YES oder den Wert TRUE haben, um diese Funktion zu aktivieren.

Sie können die Registrierdatenbankvariable **DB2_FCM_SETTINGS** mit der Option FCM_CFG_BASE_AS_FLOOR verwenden, um den Basiswert als untere Grenze für die Konfigurationsparameter *fcm_num_buffers* und *fcm_num_channels* des Datenbankmanagers festzulegen. Wenn für die Option FCM_CFG_BASE_AS_FLOOR die Einstellung YES bzw. TRUE definiert ist und diese Parameter auf AUTOMATIC gesetzt sind und einen Anfangs- oder Startwert aufweisen, werden sie von DB2 nicht auf einen Wert unterhalb dieser Grenze optimiert.

DB2_FORCE_OFFLINE_ADD_PARTITION

- Betriebssystem: Alle
- Standardwert=FALSE, Werte: FALSE oder TRUE
- Mit dieser Variablen können Sie angeben, dass Operationen zum Hinzufügen von Datenbankpartitionsservern offline ausgeführt werden. Die Standardeinstellung FALSE gibt an, dass DB2-Datenbankpartitionsserver hinzufügt werden können, ohne die Datenbank offline zu nehmen. Wenn die Operation jedoch offline ausgeführt werden soll oder wenn eine Einschränkung das Hinzufügen von Datenbankpartitionsservern im Onlinemodus der Datenbank unmöglich macht, setzen Sie **DB2_FORCE_OFFLINE_ADD_PARTITION** auf den Wert TRUE. Wenn diese Variable auf den Wert TRUE gesetzt ist, werden neue DB2-Datenbankpartitionsserver entsprechend dem Verhalten von Version 9.5 und früheren Versionen hinzugefügt. Das heißt, neue Datenbankpartitionsserver werden für die Instanz erst sichtbar, wenn sie heruntergefahren und neu gestartet wurde.

DB2_NUM_FAILOVER_NODES

- Betriebssystem: Alle
- Standardwert=2, Werte: 0 bis zur erforderlichen Anzahl der Datenbankpartitionen
- Definieren Sie **DB2_NUM_FAILOVER_NODES**, um die Anzahl der zusätzlichen Datenbankpartitionen anzugeben, die auf einer Maschine im Falle einer Funktionsübernahme gestartet werden müssen.

  In einer DB2-Datenbanklösung mit hoher Verfügbarkeit können beim Ausfall eines Datenbankservers die Datenbankpartitionen auf der Maschine, auf der der Fehler aufgetreten ist, auf einer anderen Maschine erneut gestartet werden. FCM (Fast Communication Manager) verwendet **DB2_NUM_FAILOVER_NODES**, um zu berechnen, wie viel Speicherplatz auf den verschiedenen Maschinen reserviert werden muss, um diese Funktionsübernahme zu ermöglichen.

  Betrachten Sie zum Beispiel die folgende Konfiguration:

  – Maschine A verfügt über zwei Datenbankpartitionen: 1 und 2.
  – Maschine B verfügt über zwei Datenbankpartitionen: 3 und 4.
  – Für **DB2_NUM_FAILOVER_NODES** wird sowohl auf Maschine A als auch auf Maschine B der Wert 2 angegeben.

  Bei Ausgabe des Befehls START DATABASE MANAGER reserviert FCM genügend Speicherplatz auf A und B, um bis zu vier Datenbankpartitionen zu verwalten, sodass bei Ausfall einer Maschine die beiden Datenbankpartitionen der fehlerhaften Einheit auf der anderen Maschine erneut gestartet werden können. Wenn Maschine A ausfällt, dann können die Datenbankpartitionen 1 und 2 auf Maschine B erneut gestartet wer-

den. Wenn Maschine B ausfällt, dann können die Datenbankpartitionen 3 und 4 auf Maschine A erneut gestartet werden.

DB2_PARTITIONEDLOAD_DEFAULT

- Betriebssystem: Alle unterstützten ESE-Plattformen
- Standardwert=YES, Werte: YES oder NO
- Die Registrierdatenbankvariable **DB2_PARTITIONEDLOAD_DEFAULT** ermöglicht Benutzern, die Standardfunktionsweise des Dienstprogramms LOAD in einer ESE-Umgebung zu ändern, wenn keine ESE-spezifischen LOAD-Optionen angegeben werden. Der Standardwert YES gibt an, dass in einer ESE-Umgebung, wenn Sie keine ESE-spezifischen LOAD-Optionen angeben, das Laden in allen Datenbankpartitionen versucht wird, in denen die Zieltabelle definiert ist. Wenn der Wert NO ist, wird das Laden nur in der Datenbankpartition versucht, mit der das Dienstprogramm LOAD gegenwärtig verbunden ist.

  Anmerkung: Diese Variable ist veraltet und wird in einem späteren Release möglicherweise entfernt. Der Befehl LOAD bietet verschiedene Optionen, die zur Erzielung der gleichen Funktionsweise verwendet werden können. Sie können die gleichen Ergebnisse wie mit der Einstellung NO für diese Variable erzielen, indem Sie die folgenden Optionen im Befehl **LOAD** angeben: PARTITIONED DB CONFIG MODE LOAD_ONLY OUTPUT_DBPARTNUMS x. Dabei steht x für die Partitionsnummer der Partition, in die Sie Daten laden wollen.

DB2PORTRANGE

- Betriebssystem: Windows
- Werte: nnnn:nnnn
- Dieser Wert wird auf den TCP/IP-Portbereich gesetzt, der vom FCM verwendet wird, sodass alle zusätzlichen auf einer anderen Maschine erstellten Datenbankpartitionen den gleichen Portbereich haben.

DB2 pureScaleUmgebungsvariablen

DB2_DATABASE_CF_MEMORY

- Betriebssystem: Alle
- Standardwert: 100, Werte: -1 oder 0 bis 100
- Typ: Float
- Diese Variable gilt speziell für DB2 pureScale-Umgebungen.
- **DB2_DATABASE_CF_MEMORY** wird verwendet, um den Anteil am CF-Gesamtspeicher (**CF_MEM_SZ**) anzugeben, der jeder Datenbank zugeordnet wird, für die der Datenbankkonfigurationsparameter **cf_db_mem_sz** auf AUTOMATIC eingestellt ist. Jede Datenbank, für die der Parameter **cf_db_mem_sz** auf einen bestimmten Wert eingestellt ist, ignoriert diese Registrierdatenbankvariable.
- Damit alle Datenbanken über einen gleichen Anteil der CF-Speicherressourcen verfügen können, wenn mehr als 100 Datenbanken aktiv sind, müssen für die Registrierdatenbankvariable **DB2_DATABASE_CF_MEMORY** Werte kleiner als 1 verwendet werden. Wenn beispielsweise 200 aktive Datenbanken vorhanden sind, von denen jede einen gleichen Anteil des CF-Speicherplatzes aufweist, muss diese Registrierdatenbankvariable auf den Wert 0,5 gesetzt werden.
- Die Verwendung der Registrierdatenbankvariable **DB2_DATABASE_CF_MEMORY** muss mit den Konfigurationsparametern

cf_db_mem_sz und **numdb** koordiniert werden. Weitere Informationen finden Sie im Abschnitt zur DB2 pureScale CF-Speicherparameterkonfiguration.

DB2_MCR_RECOVERY_PARALLELISM_CAP

- Betriebssystem: Alle
- Diese Variable gilt speziell für DB2 pureScale-Umgebungen.
- Wenn in Umgebungen mit mehreren Datenbanken nach dem Absturz eines Members eine Recovery erforderlich wird, wird die Anzahl der Datenbanken, die auf jedem Member parallel zueinander wiederhergestellt werden, durch den Wert des Konfigurationsparameters **numdb** oder der Registrierdatenbankvariable **DB2_MCR_RECOVERY_PARALLELISM_CAP** festgelegt, je nachdem, welcher dieser beiden Werte kleiner ist.

Abfragecompilervariablen

DB2_ANTIJOIN

- Betriebssystem: Alle
- Standardwert=NO in einer ESE-Umgebung, Standardwert=YES in einer anderen Umgebung (nicht ESE), Werte: YES, NO oder EXTEND
- Für DB2 Enterprise Server Edition: Bei Angabe von YES sucht das Optimierungsprogramm nach Möglichkeiten, „NOT EXISTS"-Unterabfragen in Antijoins umzusetzen, die von DB2 effizienter verarbeitet werden können. In anderen Umgebungen (nicht ESE): Bei Angabe von NO schränkt das Optimierungsprogramm die Möglichkeiten zur Umsetzung von „NOT EXISTS"-Unterabfragen in Antijoins ein.

  Wenn EXTEND angegeben ist, sucht das Optimierungsprogramm sowohl in ESE-Umgebungen als auch in anderen Umgebungen nach Möglichkeiten, NOT IN- und NOT EXISTS-Unterabfragen in Antijoins umzusetzen.
- Änderungen an dieser Variablen werden sofort wirksam und gelten für alle zukünftig kompilierten SQL-Anweisungen, wenn der Befehl **db2set** mit dem Parameter **-immediate** ausgegeben wird. Es ist nicht erforderlich, die Instanz erneut zu starten.

DB2_DEFERRED_PREPARE_SEMANTICS

- Betriebssystem: Alle
- Standardwert=NO, Werte: YES oder NO
- Wenn diese Registrierdatenbankvariable auf den Wert YES gesetzt ist, aktiviert sie die Semantik zur verzögerten Vorbereitung in der Weise, dass alle nicht typisierten Parametermarken, die in PREPARE-Anweisungen verwendet werden, ihre Datentypen und Längenattribute aus dem Eingabedeskriptor ableiten, der den nachfolgenden OPEN- oder EXECUTE-Anweisungen zugeordnet ist. Dadurch können nicht typisierte Parametermarken an mehr Stellen verwendet werden, als zuvor unterstützt wurden.

  Die Registrierdatenbankvariable **DB2_DEFERRED_PREPARE_SEMANTICS** muss festgelegt werden, bevor der Befehl **db2start** ausgeführt wird.

  Diese Registrierdatenbankvariable wird nur für Unicode- und SBCS-Datenbanken empfohlen.

  Anmerkung: Die Einstellung der Registrierdatenbankvariablen **DB2_DEFERRED_PREPARE_SEMANTICS** auf den Wert YES kann unerwünschte Effekte oder Ergebnisse zur Folge haben. In Fällen, in denen sich der Datentyp im Eingabedeskriptor von dem Datentyp unterscheidet, der

mit den Regeln zum Bestimmen von Datentypen nicht typisierter Ausdrücke abgeleitet wird, sind folgende Konsequenzen möglich:

– Die Abfrageleistung kann sich durch die zusätzliche Umsetzungsoperation für den Datentyp verschlechtern.
– Die Abfrage schlägt fehl, weil ein Datentyp nicht konvertiert werden kann.
– Die Abfrage kann andere Ergebnisse zurückgeben.

Nehmen Sie zum Beispiel an, dass eine Tabelle 't1' mit einer Spalte 'char_col', die als VARCHAR(10) definiert ist, die Werte '1', '100', '200', 'xxx' enthält. Ein Benutzer führt die folgende Abfrage aus:

```
select * from t1 where char_col = ?
```

Wenn der Datentyp des Eingabeparameters INTEGER ist und eine verzögerte Vorbereitung verwendet wird, wird der Datentyp der Spalte 'char_col' in einen numerischen Typ umgesetzt. Die Abfrage schlägt jedoch fehl, weil eine der Zeilen in der Tabelle einen nicht numerischen Datenwert ('xxx') enthält, der sich nicht in einen numerischen Wert konvertieren lässt.

DB2_INLIST_TO_NLJN

- Betriebssystem: Alle
- Standardwert=NO, Werte: YES oder NO
- In einigen Fällen kann der SQL- und XQuery-Compiler ein Listenvergleichselement IN in einen Join umschreiben. Betrachten Sie zum Beispiel die folgende Abfrage:

  ```
  SELECT *
   FROM EMPLOYEE
   WHERE DEPTNO IN ('D11', 'D21', 'E21')
  ```

  Diese Abfrage könnte folgendermaßen geschrieben werden:

  ```
  SELECT *
   FROM EMPLOYEE, (VALUES 'D11', 'D21', 'E21') AS V(DNO)
   WHERE DEPTNO = V.DNO
  ```

  Diese Überarbeitung könnte eine bessere Leistung erzielen, wenn es einen Index für die Spalte DEPTNO gibt. Auf die Liste von Werten würde zuerst zugegriffen und die Liste mit der Tabelle EMPLOYEE durch einen Join mit Verschachtelungsschleife mithilfe des Index zur Anwendung des Joinvergleichselements verknüpft.

  Manchmal verfügt das Optimierungsprogramm über keine genauen Informationen, um die beste Joinmethode für die umgeschriebene Version der Abfrage zu bestimmen. Dies kann der Fall sein, wenn die IN-Liste Parametermarken oder Hostvariablen enthält, die verhindern, dass das Optimierungsprogramm die Selektivität mithilfe der Katalogstatistiken ermitteln kann. Diese Registrierdatenbankvariable veranlasst das Optimierungsprogramm, Joins mit Verschachtelungsschleifen zu bevorzugen, um die Liste von Werten zu verknüpfen und dabei die Tabelle, die die IN-Liste beisteuert, als innere Tabelle im Join zu verwenden.

  Anmerkung: Wenn mindestens eine der beiden DB2-Abfragecompilervariablen **DB2_MINIMIZE_LISTPREFETCH** und **DB2_INLIST_TO_NLJN** auf YES gesetzt ist, bleibt sie auch dann aktiv, wenn REOPT(ONCE) angegeben wird.

- Änderungen an dieser Variablen werden sofort wirksam und gelten für alle zukünftig kompilierten SQL-Anweisungen, wenn der Befehl **db2set** mit dem Parameter **-immediate** ausgegeben wird. Es ist nicht erforderlich, die Instanz erneut zu starten.

DB2_LIKE_VARCHAR

- Betriebssystem: Alle
- Standardwert=Y,Y,
- Steuert die Verwendung von Statistikdaten zu Unterelementen. Dabei handelt es sich um Statistikdaten zum Inhalt von Spaltendaten, wenn diese eine Struktur in Form einer Folge von Unterfeldern oder Unterelementen aufweisen, die durch Leerzeichen getrennt sind. Die Erfassung von Statistikdaten für Unterelemente ist optional und wird durch Optionen im Befehl **RUNSTATS** oder der API gesteuert.

  Wichtig: Diese Variable ist veraltet und wird in einem zukünftigen Release möglicherweise entfernt, da Sie die Einstellungen nur in Absprache mit dem IBM Service ändern sollten.

  Diese Registrierdatenbankvariable beeinflusst, wie das Optimierungsprogramm ein Vergleichselement der folgenden Form behandelt:

  ```
  COLUMN LIKE '%xxxxxx%'
  ```

  Dabei ist xxxxxx eine beliebige Zeichenfolge.

  Die folgende Syntax zeigt, wie diese Registrierdatenbankvariable verwendet wird:

  ```
  db2set DB2_LIKE_VARCHAR=[Y|N|S|num1] [,Y|N|S|num2]
  ```

  Dabei gilt Folgendes:

  – Der Term vor dem Komma bzw. der einzige Term rechts des Vergleichselements hat folgende Bedeutung, allerdings nur, wenn für den zweiten Term der Wert N angegeben wurde oder die Spalte keine positiven Statistikdaten zu Unterelementen hat:
    - S – Das Optimierungsprogramm schätzt die Länge der einzelnen Elemente einer Folge miteinander verknüpfter Elemente, die eine Spalte bilden, ausgehend von der Länge der Zeichenfolge, die von den %-Zeichen eingeschlossen wird.
    - Y – Der Standardwert. Verwenden des Standardwerts 1,9 als Algorithmusparameter. Verwenden des Algorithmus für Unterelemente mit variabler Länge mit dem Algorithmusparameter.
    - N – Verwenden eines Algorithmus für Unterelemente mit fester Länge.
    - num1 – Verwenden des Werts von num1 als Algorithmusparameter für den Algorithmus für Unterelemente mit variabler Länge.
  – Der Term nach dem Komma hat folgende Bedeutung, jedoch nur für Spalten, die positive Statistikdaten zu Unterelementen haben:
    - N – Kein Verwenden von Statistikdaten zu Unterelementen. Der erste Term tritt in Kraft.
    - Y – Der Standardwert. Verwenden eines Algorithmus für Unterelemente mit variabler Länge, der Statistikdaten zu Unterelementen zusammen mit dem Standardwert 1,9 als Algorithmusparameter verwendet, wenn Spalten mit positiven Statistikdaten zu Unterelementen vorliegen.

- num2 – Verwenden eines Algorithmus für Unterelemente mit variabler Länge, der Statistikdaten zu Unterelementen zusammen mit dem Wert num2 als Algorithmusparameter verwendet, wenn Spalten mit positiven Statistikdaten zu Unterelementen vorliegen.

- Änderungen an dieser Variablen werden sofort wirksam und gelten für alle zukünftig kompilierten SQL-Anweisungen, wenn der Befehl **db2set** mit dem Parameter **-immediate** ausgegeben wird. Es ist nicht erforderlich, die Instanz erneut zu starten.

DB2_MINIMIZE_LISTPREFETCH

- Betriebssystem: Alle
- Standardwert=NO, Werte: YES oder NO
- Der Vorablesezugriff über Listen ist eine spezielle Zugriffsmethode auf Tabellen, bei der die den Bedingungen entsprechenden Satz-IDs (RIDs) aus einem Index abgerufen, nach Seitennummer sortiert und anschließend die Datenseiten vorab gelesen werden. Manchmal verfügt das Optimierungsprogramm über keine genauen Informationen, um zu ermitteln, ob ein Vorablesezugriff über Listen eine gute Zugriffsmethode ist. Dies kann der Fall sein, wenn die Selektivitäten von Vergleichselementen Parametermarken oder Hostvariablen enthalten, die verhindern, dass das Optimierungsprogramm die Selektivität mithilfe der Katalogstatistiken ermitteln kann.

  Diese Registrierdatenbankvariable verhindert, dass das Optimierungsprogramm in solchen Situationen einen Vorablesezugriff über Listen in Betracht zieht.

  Anmerkung: Wenn mindestens eine der beiden DB2-Abfragecompilervariablen **DB2_MINIMIZE_LISTPREFETCH** und **DB2_INLIST_TO_NLJN** auf YES gesetzt ist, bleibt sie auch dann aktiv, wenn REOPT(ONCE) angegeben wird.
- Änderungen an dieser Variablen werden sofort wirksam und gelten für alle zukünftig kompilierten SQL-Anweisungen, wenn der Befehl **db2set** mit dem Parameter **-immediate** ausgegeben wird. Es ist nicht erforderlich, die Instanz erneut zu starten.

DB2_NEW_CORR_SQ_FF

- Betriebssystem: Alle
- Standardwert=OFF, Werte: ON oder OFF
- Beeinflusst, wenn auf ON gesetzt, den Selektivitätswert, der vom Abfrageoptimierungsprogramm für bestimmte Vergleichselemente von Unterabfragen berechnet wird. Diese Variable dient zur Verbesserung der Genauigkeit des Selektivitätswerts von Gleichheitsvergleichselementen in Unterabfragen, die die Spaltenfunktion MIN oder MAX in der SELECT-Liste der Unterabfrage enthalten. Beispiel:

  ```
  SELECT * FROM T WHERE
  T.COL = (SELECT MIN(T.COL)
  FROM T WHERE ...)
  ```
- Änderungen an dieser Variablen werden sofort wirksam und gelten für alle zukünftig kompilierten SQL-Anweisungen, wenn der Befehl **db2set** mit dem Parameter **-immediate** ausgegeben wird. Es ist nicht erforderlich, die Instanz erneut zu starten.

DB2_OPT_MAX_TEMP_SIZE

- Betriebssystem: Alle

- Standardwert=NULL, Werte: Größe des Speicherbereichs in Megabyte, der von einer Abfrage in allen Tabellenbereichen für temporäre Tabellen verwendet werden kann
- Begrenzt die Größe des Speicherbereichs, den Abfragen in den temporären Tabellenbereichen nutzen können. Wenn die Variable **DB2_OPT_MAX_TEMP_SIZE** definiert wird, kann sie das Optimierungsprogramm veranlassen, einen aufwendigeren Plan als sonst auszuwählen, der jedoch weniger Speicherplatz in den Tabellenbereichen für temporäre Tabellen erfordert. Wenn Sie die Variable **DB2_OPT_MAX_TEMP_SIZE** definieren, stellen Sie sicher, dass Sie die Notwendigkeit, den Bedarf an temporärem Tabellenbereich einzuschränken, gegen die Effizienz des Plans, der durch Ihre Einstellung ausgewählt wird, geeignet abwägen.

  Wenn **DB2_WORKLOAD=SAP** definiert wird, wird die Variable **DB2_OPT_MAX_TEMP_SIZE** automatisch auf den Wert `10.240` (10 GB) gesetzt.

  Wenn Sie eine Abfrage ausführen, die über den für **DB2_OPT_MAX_TEMP_SIZE** festgelegten Wert hinaus temporären Tabellenbereich belegt, schlägt die Abfrage nicht fehl, jedoch empfangen Sie eine Warnung, dass die Leistung vielleicht nicht optimal ist, da möglicherweise nicht alle Ressourcen verfügbar sind.

  Die folgenden vom Optimierungsprogramm in Betracht gezogenen Operationen sind von dem durch **DB2_OPT_MAX_TEMP_SIZE** definierten Grenzwert betroffen:

  – Explizite Sortierungen für Operationen wie ORDER BY, DISTINCT, GROUP BY, Mischsuchjoins und Joins mit Verschachtelungsschleife
  – Explizite temporäre Tabellen
  – Implizite temporäre Tabellen für Hash-Joins und duplizierte Mischjoins
- Änderungen an dieser Variablen werden sofort wirksam und gelten für alle zukünftig kompilierten SQL-Anweisungen, wenn der Befehl **db2set** mit dem Parameter **-immediate** ausgegeben wird. Es ist nicht erforderlich, die Instanz erneut zu starten.

DB2_REDUCED_OPTIMIZATION

- Betriebssystem: Alle
- Standardwert: NO, Werte: NO, YES, beliebige Ganzzahl, DISABLE, JUMPSCAN, NO_SORT_NLJOIN oder NO_SORT_MGJOIN
- Mit dieser Registrierdatenbankvariablen können Sie entweder reduzierte Optimierungsfunktionen oder die strenge Anwendung der Optimierungsfunktionen der angegebenen Optimierungsklasse anfordern. Wenn Sie die Anzahl der verwendeten Optimierungstechniken reduzieren, können Sie dadurch die Zeit und die Ressourcenbelastung bei der Optimierung verringern.

  Beim Festlegen dieser Variablen gelten die folgenden Syntaxregeln:

  – Trennen Sie die Optionen durch ein Komma (,) und stellen Sie sicher, dass sich kein Leerzeichen vor oder nach den Kommas befindet.
  – Trennen Sie eine Option und den zugehörigen Wert durch ein einzelnes Leerzeichen.
  – Wenn die Einstellung ein Leerzeichen enthält, schließen Sie die Einstellung in Anführungszeichen (“”) ein.

  Das folgende Beispiel zeigt die richtige Syntax:

  ```
  db2set DB2_REDUCED_OPTIMIZATION="NO_SORT_NLJOIN,JUMPSCAN ON"
  ```

Anmerkung: Die Optimierungszeit und die Ressourcenbelastung wird zwar reduziert, jedoch erhöht sich das Risiko, dass ein nicht optimaler Datenzugriffsplan generiert wird. Verwenden Sie diese Registrierdatenbankvariable nur, wenn Sie von IBM oder einem IBM Partner dazu aufgefordert werden.

– Wenn der Wert NO definiert ist

  Das Optimierungsprogramm ändert seine Optimierungstechniken nicht.

– Wenn der Wert YES definiert ist

  Wenn die Optimierungsklasse 5 (Standardwert) oder eine niedrigere verwendet wird, inaktiviert das Optimierungsprogramm einige Optimierungstechniken, die möglicherweise viel Vorbereitungszeit und Ressourcen beanspruchen, jedoch in der Regel zur Generierung besserer Zugriffspläne führen.

  Wenn die Optimierungsklasse exakt 5 ist, reduziert das Optimierungsprogramm einige zusätzliche Techniken oder inaktiviert sie, wodurch sich die Optimierungszeit und die Ressourcenbelastung weiter verringern können, jedoch gleichzeitig die Gefahr wächst, dass ein nicht optimaler Zugriffsplan generiert wird. Bei Optimierungsklassen unter 5 kommen einige dieser Techniken möglicherweise ohnehin nicht zur Anwendung. Wenn sie dennoch angewandt werden, bleiben sie jedoch wirksam.

– Wenn eine beliebige Ganzzahl definiert ist

  Der Effekt ist derselbe wie bei `YES`, jedoch mit folgenden zusätzlichen Merkmalen für dynamisch vorbereitete Abfragen, die mit Klasse 5 optimiert werden. Wenn die Gesamtzahl von Joins in einem beliebigen Abfrageblock die Einstellung überschreitet, wechselt das Optimierungsprogramm zur schnellen Joinaufzählung (Greedy Join Enumeration), anstatt zusätzliche Optimierungstechniken wie zuvor für die Optimierungsklasse 5 beschrieben zu inaktivieren. Dies impliziert, dass die Abfrage mit einer ähnlichen Klasse wie Optimierungsklasse 2 optimiert wird.

– Wenn der Wert DISABLE definiert ist

  Wenn keine Einschränkung durch diese Variable **`DB2_REDUCED_OPTIMIZATION`** definiert ist, sieht die Funktionsweise des Optimierungsprogramms zuweilen vor, die Optimierung für dynamische Abfragen in Optimierungsklasse 5 dynamisch zu reduzieren. Diese Einstellung setzt diese Funktionsweise außer Kraft und zwingt das Optimierungsprogramm, die volle Optimierung der Klasse 5 durchzuführen.

– Wenn der Wert JUMPSCAN definiert ist

  Verwenden Sie diese Option, um zu steuern, ob das DB2-Optimierungsprogramm Sprungsuchoperationen verwenden kann. Sie können die folgenden Werte angeben:

  - OFF = Das DB2-Optimierungsprogramm erstellt keine Pläne mithilfe von Sprungsuchen.
  - ON = Das DB2-Optimierungsprogramm verwendet eine aufwandsbasierte Analyse, um zu ermitteln, ob Pläne generiert werden sollen, die Sprungsuchen verwenden (Standardwert).

– Wenn der Wert NO_SORT_NLJOIN definiert ist

  Das Optimierungsprogramm generiert keine Abfragepläne, die Sortierungen für Joins mit Verschachtelungsschleife (NLJN, Nested Loop

Join) erzwingen. Diese Typen von Sortierungen können zur Steigerung der Leistung nützlich sein. Daher ist bei der Verwendung der Option NO_SORT_NLJOIN Vorsicht geboten, da die Leistung erheblich beeinträchtigt werden kann.

– Wenn der Wert NO_SORT_MGJOIN definiert ist

  Das Optimierungsprogramm generiert keine Abfragepläne, die Sortierungen für Mischsuchjoins (MSJN, Merge-Scan-Joins) erzwingen. Diese Typen von Sortierungen können zur Steigerung der Leistung nützlich sein. Daher ist bei der Verwendung der Option NO_SORT_MGJOIN Vorsicht geboten, da die Leistung erheblich beeinträchtigt werden kann.

  Beachten Sie, dass die dynamische Reduzierung der Optimierung in Optimierungsklasse 5 Vorrang vor der Funktionsweise hat, die für die exakte Optimierungsklasse 5 beschrieben ist, wenn die Variable **DB2_REDUCED_OPTIMIZATION** auf den Wert YES gesetzt ist, sowie vor der Funktionsweise, die für die ganzzahlige Einstellung beschrieben ist.

- Änderungen an dieser Variablen werden sofort wirksam und gelten für alle zukünftig kompilierten SQL-Anweisungen, wenn der Befehl **db2set** mit dem Parameter **-immediate** ausgegeben wird. Es ist nicht erforderlich, die Instanz erneut zu starten.

DB2_SELECTIVITY

- Betriebssystem: Alle
- Standardwert=NO, Werte: YES oder NO
- Diese Registrierdatenbankvariable steuert, wo die Klausel SELECTIVITY in Suchbedingungen in SQL-Anweisungen verwendet werden kann.

  Wenn diese Registrierdatenbankvariable auf NO gesetzt ist, kann die Klausel SELECTIVITY nur in einem benutzerdefinierten Vergleichselement angegeben werden.

  Wenn diese Registrierdatenbankvariable auf YES gesetzt ist, kann die Klausel SELECTIVITY für die folgenden Vergleichselemente angegeben werden:

  – Ein benutzerdefiniertes Vergleichselement
  – Ein einfaches Vergleichselement, in dem mindestens ein Ausdruck Hostvariablen oder Parametermarken enthält

- Änderungen an dieser Variablen werden sofort wirksam und gelten für alle zukünftig kompilierten SQL-Anweisungen, wenn der Befehl **db2set** mit dem Parameter **-immediate** ausgegeben wird. Es ist nicht erforderlich, die Instanz erneut zu starten.

DB2_SQLROUTINE_PREPOPTS

- Betriebssystem: Alle
- Standardwert=Leere Zeichenfolge, Werte:
  – APREUSE {YES | NO}
  – BLOCKING {UNAMBIG | ALL | NO}
  – CONCURRENTACCESSRESOLUTION { USE CURRENTLY COMMITTED | WAIT FOR OUTCOME }
  – DATETIME {DEF | USA | EUR | ISO | JIS | LOC}
  – DEGREE {1 | *grad-der-parallelität* | ANY}
  – DYNAMICRULES {BIND | INVOKEBIND | DEFINEBIND | RUN | INVOKERUN | DEFINERUN}
  – EXPLAIN {NO | YES | ALL}

- EXPLSNAP {NO | YES | ALL}
- FEDERATED {NO | YES}
- INSERT {DEF | BUF}
- ISOLATION {CS | RR | UR | RS | NC}
- OPTPROFILE {profilname | schemaname.profilname}
- QUERYOPT *optimierungsgrad*
- REOPT {NONE | ONCE | ALWAYS}
- STATICREADONLY {YES|NO|INSENSITIVE}
- VALIDATE {RUN | BIND}

• Die Registrierdatenbankvariable **DB2_SQLROUTINE_PREPOPTS** kann zur Anpassung der Vorkompilierungs- und Bindeoptionen für SQL- und XQuery-Prozeduren und -funktionen verwendet werden. Wenn Sie diese Variable definieren, trennen Sie die einzelnen Optionen durch ein Leerzeichen wie im folgenden Beispiel:

```
db2set DB2_SQLROUTINE_PREPOPTS="BLOCKING ALL VALIDATE RUN"
```

Eine vollständige Beschreibung der einzelnen Optionen und ihrer Einstellungen finden Sie in den Informationen zum Befehl BIND.

Wenn Sie die gleichen Ergebnisse wie durch **DB2_SQLROUTINE_PREPOPTS** für einzelne ausgewählte Prozeduren erzielen möchten, jedoch ohne die Instanz erneut zu starten, verwenden Sie die Prozedur SET_ROUTINE_OPTS.

Leistungsvariablen

DB2_ALLOCATION_SIZE

- Betriebssystem: Alle
- Standardwert: `128 KB`, Bereich: `64 KB - 256 MB`
- Gibt die Größe von Speicherzuordnungen für Pufferpools an.

  Wenn diese Registrierdatenbankvariable auf einen höheren Wert gesetzt wird, besteht der potenzielle Vorteil darin, dass weniger Zuordnungen erforderlich sind, um eine gewünschte Größe des Speichers für einen Pufferpool zu erreichen.

  Der potenzielle Nachteil eines höheren Werts für diese Registrierdatenbankvariable besteht darin, dass Speicher verschwendet werden kann, wenn der Pufferpool um einen Wert geändert wird, der nicht einem Vielfachen der Zuordnungsgröße entspricht. Wenn zum Beispiel die Variable **DB2_ALLOCATION_SIZE** auf 8 gesetzt ist und ein Pufferpool um 4 MB verkleinert wird, werden diese 4 MB verschwendet, weil kein ganzes 8-MB-Segment freigegeben werden kann.

Anmerkung: **DB2_ALLOCATION_SIZE** ist veraltet und wird in einem späteren Release möglicherweise entfernt.

DB2_APM_PERFORMANCE

- Betriebssystem: Alle
- Standardwert: `OFF`, Werte: `ON` oder `OFF`
- Setzen Sie diese Variable auf den Wert `ON`, um die leistungsrelevanten Änderungen am Zugriffsplanmanager (APM) zu aktivieren, die sich auf die Funktionsweise des Abfragecache (Paketcache) auswirken. Diese Einstellungen werden für Produktionssysteme im Normalfall nicht empfohlen. Sie verursachen einige Einschränkungen, wie zum Beispiel die Mög-

lichkeit von Fehlern aufgrund nicht ausreichenden Paketcacheplatzes oder erhöhter Speicherauslastung (oder beider Ursachen).

Die Einstellung der Variablen **DB2_APM_PERFORMANCE** auf den Wert ON aktiviert außerdem den Modus NO PACKAGE LOCK ('Keine Paketsperre'). Dieser Modus ermöglicht den Betrieb des globalen Abfragecache ohne Verwendung von Paketsperren, bei denen es sich um interne Systemsperren handelt, die Paketeinträge im Cache davor schützen, entfernt zu werden. Der Modus NO PACKAGE LOCK kann zwar zu einer etwas verbesserten Leistung führen, jedoch sind bestimmte Datenbankoperationen nicht zulässig. Zu diesen unzulässigen Operationen können gehören: Operationen, die Pakete ungültig machen, Operationen, die Pakete funktionsunfähig machen, sowie die Operationen **PRECOMPILE**, **BIND** und **REBIND**.

DB2ASSUMEUPDATE

- Betriebssystem: Alle
- Standardwert: OFF, Werte: ON oder OFF
- Wenn diese Variable aktiviert ist, kann das DB2-Datenbanksystem annehmen, dass alle Spalten fester Länge, die in einer UPDATE-Anweisung angegeben werden, geändert werden. Dadurch muss das DB2-Datenbanksystem keinen Vergleich zwischen den vorhandenen Spaltenwerten und den angegebenen neuen Spaltenwerten durchführen, um festzustellen, ob sich die Spalte tatsächlich ändert. Wenn bei Verwendung dieser Registrierdatenbankvariablen Spalten zur Aktualisierung angegeben werden (z. B. in einer SET-Klausel), jedoch nicht wirklich geändert werden, kann dies zusätzliche Protokoll- und Indexpflegeaktivitäten zur Folge haben.

  Die Aktivierung der Registrierdatenbankvariablen **DB2ASSUMEUPDATE** erfolgt mit dem Befehl **db2start**.

DB2_AVOID_PREFETCH

- Betriebssystem: Alle
- Standardwert: OFF, Werte: ON oder OFF
- Definiert, ob ein Vorablesezugriff bei der Recovery nach einem Systemabsturz verwendet werden soll. Wenn **DB2_AVOID_PREFETCH** =ON definiert ist, wird der Vorablesezugriff nicht verwendet.

DB2BPVARS

- Betriebssystem: Wie für jeden Parameter angegeben
- Standardwert: Pfad
- Es sind zwei Gruppen von Parametern zur Optimierung von Pufferpools verfügbar. Die eine Gruppe von Parametern, die nur unter Windows zur Verfügung steht, gibt an, dass Pufferpools SCATTER-Lesezugriffe für bestimmte Typen von Containern verwenden sollen. Die andere Gruppe von Parametern, die auf allen Plattformen verfügbar ist, beeinflusst die Funktionsweise des Vorablesezugriffs.

  Wichtig: Diese Leistungsvariable gilt in Version 9.5 als veraltet und wird möglicherweise in einem zukünftigen Release entfernt. Weitere Informationen hierzu finden Sie im Abschnitt .

  Die Parameter werden in einer ASCII-Datei, jeweils ein Parameter in einer Zeile, in der Form parameter=wert angegeben. Zum Beispiel könnte eine Datei mit dem Namen bpvars.vars die folgende Zeile enthalten:

  ```
  NO_NT_SCATTER = 1
  ```

Wenn die Datei bpvars.vars beispielsweise im Pfad F:\vars\ gespeichert ist, führen Sie zur Definition dieser Variablen den folgenden Befehl aus:

```
db2set DB2BPVARS=F:\vars\bpvars.vars
```

Parameter für SCATTER-Lesezugriff

Die Parameter für den SCATTER-Lesezugriff werden für Systeme mit einem hohen Aufkommen an sequenziellen Vorableseoperationen für die jeweilige Art von Container empfohlen, für die Sie bereits den Parameter **DB2NTNOCACHE** auf ON gesetzt haben. Diese Parameter, die nur auf Windows-Plattformen verfügbar sind, heißen NT_SCATTER_DMSFILE, NT_SCATTER_DMSDEVICE und NT_SCATTER_SMS. Geben Sie den Parameter NO_NT_SCATTER an, um den SCATTER-Lesezugriff für einen Container explizit auszuschließen. Bestimmte Parameter werden dazu verwendet, den SCATTER-Lesezugriff für alle Container des angegebenen Typs zu aktivieren. Für jeden dieser Parameter ist der Standardwert 0 (bzw. OFF); die möglichen Werte sind: 0 (bzw. OFF) und 1 (bzw. ON).

Anmerkung: Sie können den SCATTER-Lesezugriff nur aktivieren, wenn **DB2NTNOCACHE** auf den Wert ON gesetzt ist, um die Windows-Dateicachefunktion abzuschalten. Wenn **DB2NTNOCACHE** auf den Wert OFF gesetzt oder nicht definiert ist, wird eine Warnung in das Protokoll mit Benachrichtigungen für die Systemverwaltung geschrieben, wenn Sie versuchen, den SCATTER-Lesezugriff für einen Container zu aktivieren. Der SCATTER-Lesezugriff bleibt in diesem Fall inaktiviert.

DB2CHKPTR

- Betriebssystem: Alle
- Standardwert: OFF, Werte: ON oder OFF
- Definiert, ob die Zeigerprüfung für Eingaben erforderlich ist oder nicht.

DB2CHKSQLDA

- Betriebssystem: Alle
- Standardwert: ON, Werte: ON oder OFF
- Definiert, ob die Prüfung von SQL-Deskriptorbereichen (SQLDA) für Eingaben erforderlich ist oder nicht.

DB2_EVALUNCOMMITTED

- Betriebssystem: Alle
- Standardwert: NO, Werte: YES, NO
- Wenn sie aktiviert ist, ermöglicht diese Variable, dass bei Suchen Zeilensperren nach Möglichkeit so lange verzögert bzw. vermieden werden, bis feststeht, dass die Daten der Auswertung der Vergleichselemente entsprechen. Wenn diese Variable aktiviert ist, findet möglicherweise auch eine Auswertung von Vergleichselementen für nicht festgeschriebene Daten statt.

  Die Variable **DB2_EVALUNCOMMITTED** ist nur anwendbar, wenn die Semantik für zurzeit festgeschriebene Daten nicht hilft, Sperrenkonflikte zu vermeiden. Wenn diese Variable definiert ist und die Semantik für zurzeit festgeschriebene Daten auf einen Suchlauf anwendbar ist, werden gelöschte Zeilen nicht übersprungen und es erfolgt keine Auswertung für Vergleichselemente für nicht festgeschriebene Daten. Stattdessen wird die zurzeit festgeschriebene Version der Zeilen und Daten verarbeitet.

  Des Weiteren ist die Variable **DB2_EVALUNCOMMITTED** nur auf Anweisungen anwendbar, die entweder mit der Isolationsstufe der Cursorstabilität oder mit der Isolationsstufe der Lesestabilität arbeiten. Darüber hinaus

werden beim Tabellensuchzugriff gelöschte Zeilen bedingungslos übersprungen, während gelöschte Schlüssel bei Indexsuchen nicht übersprungen werden, sofern nicht die Registrierdatenbankvariable **DB2_SKIPDELETED** ebenfalls definiert ist.

Die Aktivierung der Registrierdatenbankvariablen **DB2_EVALUNCOMMITTED** erfolgt mit dem Befehl **db2start**. Die Entscheidung, ob ein verzögertes Sperren anwendbar ist, wird beim Kompilieren oder Binden der Anweisung getroffen.

DB2_EXTENDED_IO_FEATURES

- Betriebssystem: AIX
- Standardwert: OFF, Werte: ON, OFF
- Setzen Sie diese Variable auf den Wert ON, um die Funktionsmerkmale zur Verbesserung der E/A-Leistung zu aktivieren. Zu diesen Verbesserungen gehören eine Erhöhung der Trefferrate von Hauptspeichercaches sowie eine Verringerung der Latenzzeit für E/A-Operationen mit hoher Priorität. Diese Funktionen sind nur für bestimmte Kombinationen aus Software- und Hardwarekonfigurationen verfügbar. Für andere Konfigurationen wird die Einstellung ON vom DB2-Datenbankverwaltungssystem oder vom Betriebssystem ignoriert. Eine Konfiguration muss die folgenden Mindestvoraussetzungen erfüllen:
  - Datenbankversion: DB2 Version 9.1
  - Eine Roheinheit (RAW) muss für Datenbankcontainer verwendet werden (Container in Dateisystemen werden nicht unterstützt).
  - Speichersubsystem: Shark DS8000 unterstützt alle erweiterten E/A-Leistungsfunktionen. Informationen zur Installation und zu den vorausgesetzten Komponenten finden Sie in der Dokumentation zum Shark DS8000.

  Die Standardeinstellungen der E/A-Priorität für HIGH, MEDIUM und LOW sind 3, 8 bzw. 12. Mithilfe der Registrierdatenbankvariablen **DB2_IO_PRIORITY_SETTING** können Sie diese Einstellungen ändern.

DB2_EXTENDED_OPTIMIZATION

- Betriebssystem: Alle
- Standardwert: OFF, Werte: ON, OFF, ENHANCED_MULTIPLE_DISTINCT oder IXOR
- Diese Variable gibt an, ob das Abfrageoptimierungsprogramm die Optimierungserweiterungen zum Verbessern der Abfrageleistung verwendet. Die Werte ON, ENHANCED_MULTIPLE_DISTINCT und IXOR geben unterschiedliche Optimierungserweiterungen an. Verwenden Sie eine durch Kommas getrennte Liste, wenn Sie mehrere Werte angeben wollen.

  Als Standardverhalten (angegeben durch die Werte OFF oder IXOR) erweitert das Optimierungsprogramm die Datenzugriffsmethode des logischen Verknüpfens von Indizes über OR (Index ORing), um OR-Vergleichselemente einzuschließen, die indizierte Spalten auch referenzieren, wenn eine nicht indizierte Spalte vorhanden ist. Betrachten Sie zum Beispiel die beiden folgenden Indexdefinitionen:

  ```
  INDEX IX2:  dept    ASC
  INDEX IX3:  job     ASC
  ```

  Die folgenden Vergleichselemente können durch die Verwendung dieser beiden Indizes erfüllt werden, wenn die Option IXOR festgelegt ist:

```
WHERE
     dept = :hv1 OR
     (job = :hv2 AND
     years >= :hv3)
```

Im Allgemeinen verbessern die **DB2_EXTENDED_OPTIMIZATION**-Einstellungen die Abfrageleistung nicht in allen Umgebungen. Einzelne Verbesserungen in der Abfrageleistung sollten durch Tests ermittelt werden.

Wichtig: Die Werte ENHANCED_MULTIPLE_DISTINCT und IXOR gelten in Version 10.1 als veraltet und werden in einem zukünftigen Release möglicherweise entfernt. Durch das Entfernen von ENHANCED_MULTIPLE_DISTINCT werden neue Erweiterungen verfügbar, die die Leistung mehrerer DISTINCT-Abfragen verbessern. Der Wert für IXOR ist redundant, da er das Standardverhalten angibt. Weitere Details finden Sie in „Registrierdatenbankvariablen mit geändertem Verhalten" in der Veröffentlichung *Neuerungen in DB2 Version 10.1*.

DB2_IO_PRIORITY_SETTING

- Betriebssystem: AIX
- Werte: HIGH:#,MEDIUM:#,LOW:# mit # aus dem Bereich von 1 bis 15
- Diese Variable wird in Kombination mit der Registrierdatenbankvariablen **DB2_EXTENDED_IO_FEATURES** verwendet. Diese Registrierdatenbankvariable bietet die Möglichkeit, die Standardeinstellungen der Werte HIGH, MEDIUM und LOW für die E/A-Priorität für das DB2-Datenbanksystem zu ändern. Die Standardwerte sind 3, 8 bzw. 12. Diese Registrierdatenbankvariable muss vor dem Start einer Instanz definiert werden. Jede Änderung erfordert einen Neustart der Instanz. Beachten Sie, dass die Einstellung dieser Registrierdatenbankvariablen allein die erweiterten E/A-Funktionen nicht aktiviert. Diese müssen mit der Variablen **DB2_EXTENDED_IO_FEATURES** aktiviert werden. Alle Systemvoraussetzungen für **DB2_EXTENDED_IO_FEATURES** gelten auch für diese Registrierdatenbankvariable.

DB2_KEEP_AS_AND_DMS_CONTAINERS_OPEN

- Betriebssystem: Alle
- Standardwert: NO, Werte: YES oder NO.
- Wenn Sie diese Variable auf ON setzen, hat jeder DMS-Tabellenbereichscontainer eine Dateikennung geöffnet, bis die Datenbank inaktiviert wird. Die Abfrageleistung kann sich erhöhen, da der Systemaufwand zum Öffnen der Container entfällt. Sie sollten diese Registrierdatenbank nur in reinen DMS-Umgebungen verwenden, andernfalls kann die Leistung der Abfragen für SMS-Tabellenbereiche negativ beeinflusst werden.

DB2_KEEPTABLELOCK

- Betriebssystem: Alle
- Standardwert: OFF, Werte: ON, TRANSACTION, OFF, CONNECTION
- Wenn diese Variable auf ON oder TRANSACTION gesetzt ist, kann das DB2-Datenbanksystem mit dieser Variablen die Tabellensperre beibehalten, wenn eine Isolationsstufe UR (Nicht festgeschriebener Lesevorgang) oder CS (Cursorstabilität) geschlossen wird. Die Tabellensperre, die beibehalten wird, wird am Ende der Transaktion freigegeben, wie dies bei Suchvorgängen mit der Isolationsstufe RS (Lesestabilität) und RR (Wiederholtes Lesen) der Fall ist.

  Wenn diese Variable auf CONNECTION gesetzt ist, wird eine Tabellensperre für eine Anwendung freigegeben, bis die Anwendung ein Rollback für

die Transaktion durchführt oder die Verbindung zurückgesetzt wird. Die Tabellensperre wird weiterhin während Commitoperationen gehalten, und Anwendungsanforderungen zum Löschen der Tabellensperre werden von der Datenbank ignoriert. Die Tabellensperre bleibt weiterhin der Anwendung zugeordnet. Wenn daher die Anwendung die Tabellensperre erneut anfordert, ist die Sperre bereits verfügbar.

Für Anwendungsauslastungen, die diese Optimierung nutzen, sollte sich die Leistung verbessern. Die Auslastungen anderer Anwendungen, die gleichzeitig ausgeführt werden, könnten jedoch beeinflusst werden. Andere Anwendungen könnten beim Zugriff auf eine vorgegebene Tabelle blockiert werden, was zu einem schlechten gemeinsamen Zugriff führt. DB2-SQL-Katalogtabellen sind von dieser Einstellung nicht betroffen. Die Einstellung CONNECTION schließt auch das Verhalten mit ein, das für die Einstellung ON oder TRANSACTION beschrieben wird.

Diese Registrierdatenbankvariable wird zum Zeitpunkt des Kompilierens oder Bindens einer Anweisung geprüft.

DB2_LARGE_PAGE_MEM

- Betriebssystem: AIX, Linux, Windows Server 2003
- Standardwert: NULL, Werte: Geben Sie mithilfe eines Sterns (*) an, dass alle gültigen Speicherbereiche die großen Seitenspeicher verwenden sollen, oder geben Sie eine durch Kommas getrennte Liste bestimmter Speicherbereiche an, die großen Seitenspeicher verwenden sollen. Die verfügbaren Bereiche variieren je nach dem verwendeten Betriebssystem. Unter AIX können die folgenden Bereiche angegeben werden: DB, DBMS, FCM, APPL oder PRIVATE. Unter Linux kann der folgende Bereich angegeben werden: DB. Unter Windows Server 2003 kann der folgende Bereich angegeben werden: DB. Die Unterstützung sehr großer Speicherseiten (so genannte Huge-Pages) ist nur unter AIX verfügbar.
- Die Registrierdatenbankvariable **DB2_LARGE_PAGE_MEM** wird dazu verwendet, die Unterstützung für große oder sehr große Speicherseiten (so genannte Huge-Pages) zu aktivieren. Durch die Definition von **DB2_LARGE_PAGE_MEM**=DB wird der Speicher mit großen Seiten für den gemeinsam genutzten Bereich des Datenbankspeichers aktiviert. Wenn der Konfigurationsparameter **database_memory** den Wert AUTOMATIC hat, wird durch diese Definition die automatische Optimierung dieses gemeinsam genutzten Speicherbereichs durch STMM inaktiviert. Unter AIX wird mit der Einstellung DB2_LARGE_PAGE_MEM=DB:16GB der Huge-Page-Speicher für den gemeinsam genutzten Bereich des Datenbankspeichers aktiviert.

  Bei Anwendungen, die durch einen intensiven Speicherzugriff gekennzeichnet sind und große Mengen an virtuellem Speicher verwenden, kann die Verwendung großer bzw. sehr großer Speicherseiten zu einer Leistungsverbesserung führen. Um das DB2-Datenbanksystem für die Verwendung großer bzw. sehr großer Speicherseiten einzurichten, müssen Sie zunächst das Betriebssystem für die Verwendung solcher Seiten konfigurieren.

  Zum Aktivieren großer Seiten für privaten Agentenspeicher unter DB2 für AIX (64 Bit) über die Einstellung **DB2_LARGE_PAGE_MEM**=PRIVATE müssen Sie das Betriebssystem für die Verwendung großer Seiten konfigurieren, und der Instanzeigner muss über die Funktionen CAP_BYPASS_RAC_VMM und CAP_PROPAGATE verfügen.

  Unter AIX 5L können Sie diese Variable auf FCM setzen. Da sich der FCM-Speicher in einer eigenen Speichergruppe befindet, müssen Sie das

Schlüsselwort FCM dem Wert der Registrierdatenbankvariablen **DB2_LARGE_PAGE_MEM** hinzufügen, um große Seiten für den FCM-Speicher zu aktivieren.

Unter Linux besteht die zusätzliche Voraussetzung, dass die Bibliothek *libcap.so.1* verfügbar sein muss. Diese Bibliothek muss installiert sein, damit diese Option funktioniert. Wenn diese Option aktiviert wird und die Bibliothek im System nicht vorhanden ist, inaktiviert die DB2-Datenbank die großen Kernelseiten und setzt die Verarbeitung wie ohne sie fort.

Zum Prüfen, ob unter Linux große Kernelseiten verfügbar sind, können Sie den folgenden Befehl absetzen:

```
cat /proc/meminfo
```

Wenn große Kernelseiten verfügbar sind, sollten die drei folgenden Zeilen (mit je nach Größe des auf Ihrem Server konfigurierten Speichers unterschiedlichen Werten) angezeigt werden:

```
HugePages_Total:   200
HugePages_Free:    200
Hugepagesize:    16384 kB
```

Wenn diese Zeilen nicht angezeigt werden oder `HugePages_Total` den Wert 0 hat, ist eine Konfiguration des Betriebssystems oder des Kernels erforderlich.

OUnter Windows ist die Menge an Speicher mit großen Seiten, die auf dem System verfügbar ist, kleiner als der insgesamt verfügbare Speicher. Wenn das System einige Zeit aktiv war, kann der Speicher fragmentiert werden, sodass die Menge an Speicher mit großen Seiten abnimmt. Für die Registrierdatenbankvariable **DB2_ALLOCATION_SIZE** sollte ein hoher Wert eingestellt werden, wie z. B. 256 Megabyte, um beim Zuordnen von großen Speicherseiten unter Windows eine konsistente Leistung zu erzielen. (Beachten Sie, dass Sie für **DB2_ALLOCATION_SIZE** die Instanz stoppen und erneut starten müssen, damit die Änderungen wirksam werden.)

DB2_LOGGER_NON_BUFFERED_IO

- Betriebssystem: Alle
- Standardwert: `AUTOMATIC`, Werte: `AUTOMATIC`, `ON` oder `OFF`
- Mit dieser Variablen können Sie steuern, ob direkte E/A (DIO) im Protokolldateisystem verwendet wird. Wenn die Variable **DB2_LOGGER_NON_BUFFERED_IO** auf den Wert `AUTOMATIC` gesetzt wird, werden aktive Protokolldateien (nämlich die primären Protokolldateien) mit direkter Ein-/Ausgabe (DIO) geöffnet und alle anderen Dateien der Protokollfunktion werden gepuffert. Wenn sie auf den Wert `ON` gesetzt ist, werden alle Protokolldateihandles mit DIO geöffnet. Wenn sie auf den Wert `OFF` gesetzt ist, werden alle Protokolldateihandles gepuffert.

DB2MAXFSCRSEARCH

- Betriebssystem: Alle
- Standardwert: `5`, Werte: `-1`, `1` bis `33.554`
- Gibt die Anzahl der Steuersätze für freien Speicherbereich (FSCRs - Free Space Control Records) an, die beim Einfügen eines Eintrags in eine Tabelle zu durchsuchen sind. Standardmäßig werden fünf Steuersätze für freien Speicherbereich durchsucht. Mit diesem Wert können Sie die Gewichtung zwischen Einfügegeschwindigkeit und Speicherwiederverwendung beeinflussen. Mit hohen Werten optimieren Sie die Speicherwiederverwendung. Mit niedrigen Werten optimieren Sie die Einfügegeschwindigkeit. Der Wert `-1` zwingt den Datenbankmanager, alle Steuersätze für freien Speicherbereich zu durchsuchen.

DB2_MAX_INACT_STMTS

- Betriebssystem: Alle
- Standardwert: nicht definiert, Werte: bis zu 4.000.000.000
- Diese Variable überschreibt den Standardgrenzwert für die Anzahl inaktiver Anweisungen, die jeweils von einer Anwendung behalten werden. Sie können einen anderen Wert auswählen, um die Größe des Systemmonitorzwischenspeichers zu erhöhen bzw. zu verringern, der für Informationen inaktiver Anweisungen verwendet wird. Der Standardgrenzwert ist 250.

  Die Kapazität des Systemmonitorzwischenspeichers kann sich erschöpfen, wenn eine Anwendung eine sehr große Anzahl von Anweisungen in einer UOW (Unit of Work, Arbeitseinheit) enthält oder wenn eine große Anzahl von Anwendungen gleichzeitig ausgeführt wird.

DB2_MAX_NON_TABLE_LOCKS

- Betriebssystem: Alle
- Standardwert: YES, Werte: Siehe Beschreibung
- Diese Variable definiert die maximale Anzahl von NON-Tabellensperren, die eine Transaktion haben kann, bevor sie alle diese Sperren freigibt. NON-Tabellensperren sind Tabellensperren, die in der Hashtabelle und der Transaktionskette verbleiben, selbst wenn sie von der Transaktion nicht mehr verwendet werden. Da Transaktionen häufig mehrmals auf die gleiche Tabelle zugreifen, kann das Beibehalten der Sperren und das Ändern ihres Status in NON die Leistung verbessern.

  Im Hinblick auf optimale Ergebnisse entspricht der für diese Variable empfohlene Wert der maximalen Anzahl von Tabellen, auf die voraussichtlich durch eine beliebige Verbindung zugegriffen wird. Wenn kein benutzerdefinierter Wert angegeben wird, gilt folgender Standardwert: Wenn die Sperrenliste (**locklist**) größer oder gleich

  SQLP_THRESHOLD_VAL_OF_LRG_LOCKLIST_SZ_FOR_MAX_NON_LOCKS

  (zurzeit 8000) ist, ist der Standardwert

  SQLP_DEFAULT_MAX_NON_TABLE_LOCKS_LARGE

  (zurzeit 150). Ansonsten ist der Standardwert

  SQLP_DEFAULT_MAX_NON_TABLE_LOCKS_SMALL

  (zurzeit 0).

DB2_MDC_ROLLOUT

- Betriebssystem: Alle
- Standardwert: IMMEDIATE, Werte: IMMEDIATE, OFF oder DEFER
- Mit dieser Variablen wird eine Leistungsverbesserung für Löschvorgänge in MDC-Tabellen aktiviert, die als „Rollout" bezeichnet wird. Ein Rollout ist eine schnellere Möglichkeit, Zeilen in einer MDC-Tabelle zu löschen, wenn ganze Zellen (Schnittpunkte von Dimensionswerten) in einer Suchanweisung mit DELETE gelöscht werden. Die Vorteile sind eine reduzierte Protokollierung und eine effizientere Verarbeitung.
- Die Einstellung der Variablen kann drei mögliche Ergebnisse haben:
  - Kein Rollout - wenn der Wert OFF angegeben wird.
  - Sofortiger Rollout - wenn der Wert IMMEDIATE angegeben wird.
  - Rollout mit verzögerter Indexbereinigung - wenn der Wert DEFER angegeben wird.

- Wenn der Wert nach dem Start geändert wird, berücksichtigen alle neuen Kompilierungen einer Anweisung die neue Einstellung der Registrierdatenbankvariablen. Bei Anweisungen, die sich im Paketcache befinden, erfolgt die Änderung in der Löschverarbeitung erst, wenn die Anweisung erneut kompiliert wird. Die Anweisung SET CURRENT MDC ROLLOUT MODE überschreibt den Wert der Variablen **DB2_MDC_ROLLOUT** auf der Anwendungsverbindungsebene.
- In DB2 Version 9.7 und späteren Releases wird der Wert DEFER für eine datenpartitionierte MDC-Tabelle mit partitionierten Satz-ID-Indizes nicht unterstützt. Es werden nur die Werte OFF und IMMEDIATE unterstützt. Der Modus des Rollouts mit Bereinigung ist IMMEDIATE, wenn die Registrierdatenbankvariable **DB2_MDC_ROLLOUT** auf den Wert DEFER gesetzt ist oder wenn das Sonderregister CURRENT MDC ROLLOUT MODE auf den Wert DEFERRED gesetzt ist, um die Einstellung der Variablen **DB2_MDC_ROLLOUT** zu überschreiben.

  Wenn nur nicht partitionierte Satz-ID-Indizes für die MDC-Tabelle vorhanden sind, wird ein Rollout mit verzögerter Indexbereinigung unterstützt.
- Änderungen an dieser Variable werden sofort wirksam und gelten für alle zukünftigen kompilierten SQL-Anweisungen. Es ist nicht nötig, die Instanz erneut zu starten oder den Befehl **db2set** mit dem Parameter **-immediate** abzusetzen.

DB2MEMDISCLAIM

- Betriebssystem: Alle
- Standardwert: YES, Werte: YES oder NO
- Der von den Prozessen des DB2-Datenbanksystems verwendete Speicher verfügt möglicherweise über einen zugeordneten Paging-Bereich. Die Reservierung dieses Paging-Bereichs bleibt möglicherweise auch dann bestehen, wenn der zugehörige Speicher freigegeben wird. Ob dies der Fall ist, hängt von der (optimierbaren) Richtlinie für die Speicherzuordnung der Verwaltung des virtuellen Speichers des Betriebssystems ab. Die Registrierdatenbankvariable **DB2MEMDISCLAIM** steuert, ob DB2-Agenten explizit anfordern, dass das Betriebssystem die Zuordnung des reservierten Paging-Bereichs zu dem freigegebenen Speicher aufhebt.

  Wenn **DB2MEMDISCLAIM** auf YES gesetzt wird, sinkt der Bedarf an Paging-Bereich und möglicherweise auch die auf Seitenwechsel (Paging) zurückzuführende Plattenaktivität. Wenn **DB2MEMDISCLAIM** auf NO gesetzt wird, steigt der Bedarf an Paging-Bereich und möglicherweise auch die auf Seitenwechsel zurückzuführende Plattenaktivität. In einigen Fällen, in denen reichlich Paging-Bereich und so viel Realspeicher verfügbar ist, dass keine Seitenwechsel auftreten, bietet die Einstellung NO eine geringfügige Leistungssteigerung.

DB2_MEM_TUNING_RANGE

- Betriebssystem: Alle
- Standardwert: NULL, Werte: eine Folge von Prozentsätzen *n*, *m* mit*n=minfree* und *m=maxfree*
- Die Größe des physischen Speichers, der von der DB2-Instanz nicht beansprucht wird, spielt eine wichtige Rolle, weil durch sie festgelegt wird, wie viel Speicher andere Anwendungen, die auf derselben Maschine aktiv sind, nutzen können. Wenn die automatische Optimierung für den gemeinsam genutzten Speicher aktiviert ist, hängt die Größe des physischen Speichers, der von einer bestimmten Instanz nicht genutzt wird,

vom Speicherbedarf der Instanz (und ihrer aktiven Datenbanken) ab. Wenn eine Instanz dringend zusätzlichen Speichers bedarf, ordnet sie Speicher zu, bis der freie physische Speicher auf dem System den Prozentsatz erreicht, der durch den Parameter *minfree* angegeben wird. Wenn die Instanz weniger Speicher bedarf, behält sie eine größere Kapazität an freiem physischen Speicher bei, die als Prozentsatz durch den Parameter *maxfree* angegeben wird. Daher ist es erforderlich, dass der für *minfree* definierte Wert kleiner als der Wert für *maxfree* ist.

Wenn diese Variable nicht festgelegt wird, berechnet der DB2-Datenbankmanager Werte für *minfree* und *maxfree* auf der Basis der Speicherkapazität auf dem Server. Die Einstellung dieser Variablen hat keine Wirkung, sofern Sie nicht mit aktiviertem Self-Tuning Memory Manager (STMM) arbeiten und den Parameter **database_memory** auf den Wert AUTOMATIC gesetzt haben.

- Änderungen an dieser Variable werden sofort wirksam und gelten für alle zukünftigen kompilierten SQL-Anweisungen. Es ist nicht nötig, die Instanz erneut zu starten oder den Befehl **db2set** mit dem Parameter **-immediate** abzusetzen.

DB2_MMAP_READ

- Betriebssystem: AIX
- Standardwert: OFF, Werte: ON oder OFF
- Diese Variable wird in Verbindung mit **DB2_MMAP_WRITE** verwendet, damit das DB2-Datenbanksystem 'mmap' als alternative E/A-Methode verwenden kann.

  Wenn diese Variablen auf ON gesetzt sind, umgehen Daten, die in die DB2-Pufferpools geschrieben und daraus gelesen werden, den AIX-Speichercache und verwenden vom Hauptspeicher zugeordnete Ein-/Ausgabe. Wenn der DB2-Pufferpool relativ klein ist und Sie diesen Pufferpool nicht vergrößern können oder wollen, sollten Sie die Nutzung des AIX-Speichercache in Erwägung ziehen, indem Sie die Variablen **DB2_MMAP_READ** und **DB2_MMAP_WRITE** auf OFF setzen.

DB2_MMAP_WRITE

- Betriebssystem: AIX
- Standardwert: OFF, Werte: ON oder OFF
- Diese Variable wird in Verbindung mit **DB2_MMAP_READ** verwendet, damit das DB2-Datenbanksystem 'mmap' als alternative E/A-Methode verwenden kann.

  Wenn diese Variablen auf ON gesetzt sind, umgehen Daten, die in die DB2-Pufferpools geschrieben und daraus gelesen werden, den AIX-Speichercache und verwenden vom Hauptspeicher zugeordnete Ein-/Ausgabe. Wenn der DB2-Pufferpool relativ klein ist und Sie diesen Pufferpool nicht vergrößern können oder wollen, sollten Sie die Nutzung des AIX-Speichercache in Erwägung ziehen, indem Sie die Variablen **DB2_MMAP_READ** und **DB2_MMAP_WRITE** auf OFF setzen.

DB2_NO_FORK_CHECK

- Betriebssystem: UNIX
- Standardwert: OFF, Werte: ON oder OFF
- Wenn diese Variable aktiviert ist, minimiert der DB2-Laufzeitclient Prüfungen zur Bestimmung, ob der aktuelle Prozess ein Ergebnis eines fork-Aufrufs ist. Dies kann die Leistung von DB2-Anwendungen erhöhen, die nicht mit der API fork() arbeiten.

DB2NTMEMSIZE

- Betriebssystem: Windows
- Standardwert: je nach Speichersegment unterschiedlich
- Windows erfordert, dass alle gemeinsam genutzten Speichersegmente während der DLL-Initialisierung reserviert werden, um übereinstimmende Adressen in allen Prozessen zu gewährleisten. **DB2NTMEMSIZE** ermöglicht Benutzern das Überschreiben der DB2-Standardwerte unter Windows (falls erforderlich). In den meisten Situationen sollte der Standardwert ausreichen. Folgende Speichersegmente, Standardgrößen und Überschreibungsoptionen sind verfügbar:
  1. Parallele FCM-Puffer: Standardgröße ist 512 MB auf 32-Bit-Plattformen; 4,5 GB auf 64-Bit-Plattformen; Überschreibungsoption ist FCM:*anzahl_byte*
  2. Kommunikation im abgeschirmten Modus: Standardgröße ist 80 MB auf 32-Bit-Plattformen; 512 MB auf 64-Bit-Plattformen; Überschreibungsoption ist APLD:*anzahl_byte*

  Sie können mehrere Segmente überschreiben, indem Sie die Überschreibungsoptionen durch ein Semikolon (;) voneinander trennen. Wenn Sie z. B. die FCM-Puffer in einer 32-Bit-Version von DB2 auf 1 GB und die abgeschirmten gespeicherten Prozeduren auf 256 MB begrenzen wollen, geben Sie Folgendes an:

  ```
  db2set DB2NTMEMSIZE=FCM:1073741824;APLD:268435456
  ```

DB2NTNOCACHE

- Betriebssystem: Windows
- Standardwert: OFF, Werte: ON oder OFF
- Die Registrierdatenbankvariable **DB2NTNOCACHE** gibt an, ob das DB2-Datenbanksystem Datenbankdateien mit der Option NOCACHE öffnet. Wenn **DB2NTNOCACHE** auf ON gesetzt ist, wird das Zwischenspeichern im Dateisystemcache inaktiviert. Wenn **DB2NTNOCACHE** auf OFF gesetzt ist, speichert das Betriebssystem DB2-Dateien im Cache. Dies gilt für alle Daten außer für Dateien, die Langfelddaten oder LOB-Daten enthalten. Durch Inaktivieren der Cachefunktion des Betriebssystems steht mehr Speicher für die Datenbank zur Verfügung, sodass der Pufferpool oder der Sortierspeicher vergrößert werden kann.

  Unter Windows werden Dateien im Cache zwischengespeichert, sobald sie geöffnet werden. Dies ist das Standardverhalten. Für jedes GB in der Datei wird aus einem Systempool ein MB reserviert. Verwenden Sie diese Registrierdatenbankvariable, um die nicht dokumentierte Begrenzung von 192 MB für den Cache außer Kraft zu setzen. Wenn die Cachebegrenzung erreicht ist, wird ein Fehler wegen nicht ausreichender Ressource ausgegeben.
- Änderungen an dieser Variable werden sofort wirksam und gelten für alle zukünftigen kompilierten SQL-Anweisungen. Es ist nicht nötig, die Instanz erneut zu starten oder den Befehl **db2set** mit dem Parameter **-immediate** abzusetzen.

Anmerkung: Bei Tabellenbereichscontainern ermöglicht die Klausel NO FILE SYSTEM CACHING bei Anweisung ALTER TABLESPACE oder CREATE TABLESPACE dieselben Vorteile, die sich ergeben, wenn **DB2NTNOCACHE** auf ON gesetzt wird.

DB2NTPRICLASS

- Betriebssystem: Windows

- Standardwert: NULL, Werte: R, H, (beliebiger anderer Wert)
- Stellt die Prioritätsklasse für die DB2-Instanz (Programm DB2SYSCS.EXE) ein. Es gibt drei Prioritätsklassen:
  - NORMAL_PRIORITY_CLASS (die Standardprioritätsklasse)
  - REALTIME_PRIORITY_CLASS (wird durch R definiert)
  - HIGH_PRIORITY_CLASS (wird durch H definiert)

  Diese Variable wird in Verbindung mit einzelnen Threadprioritäten (mit **DB2PRIORITIES** eingestellt) verwendet, um die absolute Priorität von DB2-Threads relativ zu anderen Threads im System zu bestimmen.

  Anmerkung: Die Variable **DB2NTPRICLASS** ist veraltet und sollte nur auf Empfehlung des Kundendienstes verwendet werden. Verwenden Sie DB2-Serviceklassen zur Anpassung der Agentenpriorität und der Vorableseprioritä t. Bei Verwendung dieser Variablen ist Sorgfalt geboten. Eine falsche Verwendung kann sich negativ auf die Gesamtleistung des Systems auswirken.

  Weitere Informationen finden Sie unter der API SetPriorityClass() in der Win32-Dokumentation.

DB2NTWORKSET

- Betriebssystem: Windows
- Standardwert: 1,1
- Dient zur Änderung der minimalen und der maximalen Größe des Arbeitsbereichs, der für den DB2-Datenbankmanager verfügbar ist. Standardmäßig kann der Arbeitsbereich eines Prozesses, wenn unter Windows keine Seitenwechsel (Paging) auftreten, nach Bedarf wachsen. Wenn es jedoch zum Paging kommt, liegt der maximale Arbeitsbereich, den ein Prozess haben kann, bei annähernd 1 MB. Mit **DB2NTWORKSET** kann dieser Standardwert überschrieben werden.

  Geben Sie **DB2NTWORKSET** in der Syntax **DB2NTWORKSET**=*min*,*max* an, wobei *min* und *max* in Megabyte angegeben werden.

DB2_OVERRIDE_BPF

- Betriebssystem: Alle
- Standardwert: nicht definiert, Werte: eine positive Anzahl von Seiten ODER <eintrag>[;<eintrag>...] mit <eintrag>=<pufferpool-ID>,<anzahl der seiten>
- Diese Variable gibt die Größe des Pufferpools (in Seiten) an, der bei der Aktivierung der Datenbank, bei der aktualisierenden Recovery oder bei der Recovery nach einem Systemabsturz zu erstellen ist. Diese Variable ist nützlich, wenn es während der Datenbankaktivierung, der aktualisierenden Recovery oder der Recovery nach einem Systemabsturz aufgrund begrenzter Speicherkapazitäten zu Fehlern kommt. Die Speicherknappheit kann infolge des seltenen Falls eines Mangels an Realspeicher auftreten oder durch den Versuch seitens des Datenbankmanagers verursacht werden, einen großen Pufferpool für nicht korrekt konfigurierte Pufferpools zuzuordnen. Wenn z. B. selbst ein Pufferpool der Minimalgröße von 16 Seiten nicht vom Datenbankmanager bereitgestellt werden kann, haben Sie die Möglichkeit, mithilfe dieser Umgebungsvariablen eine kleinere Anzahl von Seiten anzugeben. Der für diese Variable angegebene Wert überschreibt die aktuelle Pufferpoolgröße.

Sie können auch <eintrag>[;<eintrag>...] (mit <eintrag>=<pufferpool-ID>,<anzahl der seiten>) angeben, um die Größe aller oder einer Teilmenge der Pufferpools temporär zu ändern, sodass sie gestartet werden können.

DB2_PINNED_BP

- Betriebssystem: AIX, HP-UX, Linux
- Standardwert: NO, Werte: YES oder NO.
- Die Einstellung dieser Variablen auf den Wert YES bewirkt, dass DB2 anfordert, dass das Betriebssystem den gemeinsam genutzten Datenbankspeicher von DB2 im Hauptspeicher fixiert und dort belässt. Wenn DB2 so konfiguriert wird, dass der gemeinsam genutzte Datenbankspeicher im Hauptspeicher behalten wird, muss sorgfältig sichergestellt werden, dass das System nicht überlastet wird, da das Betriebssystem die Speicherkapazitäten nun weniger flexibel verwalten kann.

  Unter Linux ist neben einer Modifizierung dieser Registrierdatenbankvariablen auch die Bibliothek libcap.so.1 erforderlich.

  Wenn diese Variable auf YES gesetzt wird, kann die automatische Optimierung für den gemeinsam genutzten Datenbankspeicher (aktiviert durch die Einstellung AUTOMATIC für den Konfigurationsparameter **database_memory**) nicht aktiviert werden.

  Für HP-UX in einer 64-Bit-Umgebung müssen Sie zusätzlich zur Änderung dieser Registrierdatenbankvariablen der DB2-Instanzgruppe das Zugriffrecht MLOCK erteilen. Dazu führt ein Benutzer mit Rootberechtigung die folgenden Aktionen aus:

  1. Er fügt die DB2-Instanzgruppe der Datei /etc/privgroup hinzu. Wenn die DB2-Instanzgruppe beispielsweise zur Gruppe db2iadm1 gehört, muss der Datei /etc/privgroup die folgende Zeile hinzugefügt werden:

     ```
     db2iadm1 MLOCK
     ```

  2. Er führt den folgenden Befehl aus:

     ```
     setprivgrp -f /etc/privgroup
     ```

DB2PRIORITIES

- Betriebssystem: Alle
- Einstellung der Werte von der Plattform abhängig
- Steuert die Prioritäten von DB2-Prozessen und -Threads.

Anmerkung: Die Variable **DB2PRIORITIES** ist veraltet und sollte nur auf Empfehlung des Kundendienstes verwendet werden. Verwenden Sie DB2-Serviceklassen zur Anpassung der Agentenpriorität und der Vorableseprioritär.

DB2_RCT_FEATURES

- Betriebssystem: Alle
- Standardwert: NULL, Werte: GROUPUPDATE=[ON|OFF]. Der Standardwert für GROUPUPDATE lautet OFF.
- Diese Variable ermöglicht eine optimierte und reduzierte Aktualierungsverarbeitung einer Anweisung UPDATE mit Suche, deren Ziel mehrere Zeilen in einer Bereichsclustertabelle sind, wenn für die führende Spalte und die Teilmengen von Schlüsselsortierfolgen nur gleiche Vergleichselemente angegeben sind. Auch die Protokollierung ist reduziert, da statt einem Protokolleintrag pro aktualisierte Zeile für alle auf einer Seite aktualisierte Zeilen nur ein einziger Protokolleintrag vorgenommen wird.

Verwendung:

```
db2set DB2_RCT_FEATURES=GROUPUPDATE=ON
```

DB2_RESOURCE_POLICY

- Betriebssystem: AIX 5 oder höher, alle Linux-Versionen außer zSeries (32-Bit), Windows Server 2003 oder höher
- Standardwert: nicht definiert. Werte: gültiger Pfad zur Konfigurationsdatei. AUTOMATIC auf POWER7-Systemen, auf denen AIX 6.1 Technology Level (TL) 5 oder höher ausgeführt wird.
- Definiert eine Ressourcenrichtlinie, mit deren Hilfe die von der DB2-Datenbank genutzten Betriebssystemressourcen begrenzt werden können, oder sie enthält Regeln für die Zuordnung bestimmter Betriebssystemressourcen zu bestimmten DB2-Datenbankobjekten. Unter AIX, Linux oder Windows kann diese Registrierdatenbankvariable zum Beispiel zur Begrenzung der vom DB2-Datenbanksystem genutzten Gruppe von Prozessoren verwendet werden. Der Speicherbereich der Ressourcensteuerung variiert in Abhängigkeit vom Betriebssystem.

  Auf Maschinen unter AIX und Linux mit aktiviertem NUMA kann eine Richtlinie definiert werden, die angibt, welche Ressourcengruppen das DB2-Datenbanksystem verwendet. Wenn die Ressourcengruppenbindung verwendet wird, wird jeder einzelne DB2-Prozess an eine bestimmte Ressourcengruppe gebunden. Dies kann in einigen Leistungsoptimierungsszenarios von Vorteil sein.

  Auf POWER7-Systemen, auf denen AIX 6.1 Technology Level (TL) 5 oder höher ausgeführt wird, kann diese Variable auf den Wert AUTOMATIC gesetzt werden. Mit dieser Einstellung ermittelt das DB2-Datenbanksystem automatisch die Hardwaretopologie und weist den verschiedenen Hardwaremodulen Engine-Dispatchable-Units (EDUs) auf eine Weise zu, mit der Speicherplatz von mehreren EDUs, die auf dieselben Speicherbereiche zugreifen müssen, effizienter gemeinsam genutzt werden kann. Diese Einstellung ist für POWER7-Systeme mit 16 oder mehr Kernen konzipiert und kann bei einigen Workloads zu einer verbesserten Abfrageleistung führen. Am besten ist es, vor und nach der Einstellung des Werts AUTOMATIC für diese Variable eine Leistungsanalyse der Workload durchzuführen, um Leistungsverbesserungen zu prüfen.

  Sie können die Registrierdatenbankvariable festlegen, um den Pfad zu einer Konfigurationsdatei anzugeben, die eine Richtlinie für das Binden von DB2-Prozessen an Betriebssystemressourcen definiert. Die Ressourcenrichtlinie ermöglicht Ihnen die Angabe einer Gruppe von Betriebssystemressourcen, auf die das DB2-Datenbanksystem zu beschränken ist. Jeder DB2-Prozess wird an eine einzige Ressource der Gruppe gebunden. Die Ressourcenzuordnung erfolgt in einem zirkulären Umlauf.

  Beispiele für Konfigurationsdateien:

  Beispiel 1: Binden aller DB2-Prozesse entweder an CPU 1 oder an CPU 3.

```
<RESOURCE_POLICY>
  <GLOBAL_RESOURCE_POLICY>
   <METHOD>CPU</METHOD>
   <RESOURCE_BINDING>
    <RESOURCE>1</RESOURCE>
   </RESOURCE_BINDING>
   <RESOURCE_BINDING>
    <RESOURCE>3</RESOURCE>
   </RESOURCE_BINDING>
  </GLOBAL_RESOURCE_POLICY>
 </RESOURCE_POLICY>
```

Beispiel 2: (nur AIX) Binden von DB2-Prozessen an eine der folgenden Ressourcengruppen: sys/node.03.00000, sys/node.03.00001, sys/node.03.00002, sys/node.03.00003

```
<RESOURCE_POLICY>
  <GLOBAL_RESOURCE_POLICY>
   <METHOD>RSET</METHOD>
   <RESOURCE_BINDING>
    <RESOURCE>sys/node.03.00000</RESOURCE>
   </RESOURCE_BINDING>
   <RESOURCE_BINDING>
    <RESOURCE>sys/node.03.00001</RESOURCE>
   </RESOURCE_BINDING>
   <RESOURCE_BINDING>
    <RESOURCE>sys/node.03.00002</RESOURCE>
   </RESOURCE_BINDING>
   <RESOURCE_BINDING>
    <RESOURCE>sys/node.03.00003</RESOURCE>
   </RESOURCE_BINDING>
  </GLOBAL_RESOURCE_POLICY>
 </RESOURCE_POLICY>
```

Anmerkung (nur AIX): Die Verwendung der Methode RSET setzt die CAP_NUMA_ATTACH-Funktionalität voraus.

Beispiel 3 (nur Linux): Binden des gesamten Speichers aus den Pufferpool-IDs 2 und 3, die der Datenbank SAMPLE zugeordnet sind, an NUMA-Knoten 3. Darüber hinaus: Verwenden von 80 % des gesamten Datenbankspeichers für das Binden an NUMA-Knoten 3 sowie von 20 % zum einheitenübergreifenden Verteilen (Striping) über alle Knoten für nicht pufferpoolspezifischen Speicher:

```
<RESOURCE_POLICY>
  <DATABASE_RESOURCE_POLICY>
   <DBNAME>sample</DBNAME>
   <METHOD>NODEMASK</METHOD>
   <RESOURCE_BINDING>
    <RESOURCE>3</RESOURCE>
    <DBMEM_PERCENTAGE>80</DBMEM_PERCENTAGE>
    <BUFFERPOOL_BINDING>
     <BUFFERPOOL_ID>2</BUFFERPOOL_ID>
     <BUFFERPOOL_ID>3</BUFFERPOOL_ID>
    </BUFFERPOOL_BINDING>
   </RESOURCE_BINDING>
  </DATABASE_RESOURCE_POLICY>
 </RESOURCE_POLICY>
```

Beispiel 4 (nur für Linux und Windows): Definieren zweier voneinander unabhängiger Prozessorgruppen (Processor Sets), die durch die CPU-Masken 0x0F und 0xF0 angegeben werden. Binden von DB2-Prozessen und Pufferpool-ID 2 an die Prozessergruppe 0x0F und von DB2-Prozessen und Pufferpool-ID 3 an die Prozessorgruppe 0xF0. Verwenden von 50 % des gesamten Datenbankspeichers für die Bindung jeder Prozessorgruppe.

Diese Ressourcenrichtlinie ist nützlich, wenn eine Zuordnung zwischen Prozessoren und NUMA-Knoten gewünscht wird. Ein Beispiel für solch ein Szenario ist ein System mit 8 Prozessoren und 2 NUMA-Knoten, bei dem die Prozessoren 0 bis 3 zum NUMA-Knoten 0 und die Prozessoren 4 bis 7 zum NUMA-Knoten 1 gehören. Diese Ressourcenrichtlinie ermöglicht eine Prozessorbindung, indem sie gleichzeitig die Speicherlokalität (d. h. einen Hybrid der CPU-Methode und der NODEMASK-Methode) beibehält.

```
 <RESOURECE_POLICY>
   <DATABASE_RESOURCE_POLICY>
     <DBNAME>sample</DBNAME>
```

```
    <METHOD>CPUMASK</METHOD>
    <RESOURCE_BINDING>
      <RESOURCE>0x0F</RESOURCE>
      <DBMEM_PERCENTAGE>50</DBMEM_PERCENTAGE>
      <BUFFERPOOL_BINDING>
        <BUFFERPOOL_ID>2</BUFFERPOOL_ID>
      </BUFFERPOOL_BINDING>
    </RESOURCE_BINDING>
    <RESOURCE_BINDING>
      <RESOURCE>0x0F</RESOURCE>
      <DBMEM_PERCENTAGE>50</DBMEM_PERCENTAGE>
      <BUFFERPOOL_BINDING>
        <BUFFERPOOL_ID>3</BUFFERPOOL_ID>
      </BUFFERPOOL_BINDING>
    </RESOURCE_BINDING>
  </DATABASE_RESOURCE_POLICY>
</RESOURCE_POLICY>
```

Anmerkung: Die Verwendung der Methode RSET setzt die CAP_NUMA_ATTACH-Funktionalität voraus und wird unter Linux nicht unterstützt.

Die in der Registrierdatenbankvariablen **DB2_RESOURCE_POLICY** angegebene Konfigurationsdatei akzeptiert ein SCHEDULING_POLICY-Element. Über das SCHEDULING_POLICY-Element können Sie auf einigen Plattformen Folgendes auswählen:

– Die Planungsrichtlinie des Betriebssystems, die vom DB2-Server verwendet wird

  Sie können eine Betriebssystemplanungsrichtlinie für DB2 unter AIX und für DB2 unter Windows über die Registrierdatenbankvariable **DB2NTPRICLASS** definieren.

– Die Betriebssystemprioritäten, die von einzelnen DB2-Serveragenten verwendet werden

Alternativ können Sie zur Steuerung der Betriebssystemplanungsrichtlinie und zur Festlegung der DB2-Agentenprioritäten die Registrierdatenbankvariablen **DB2PRIORITIES** und **DB2NTPRICLASS** verwenden. Die Spezifikation eines SCHEDULING_POLICY-Elements in der Konfigurationsdatei für Ressourcenrichtlinien stellt eine gemeinsame Position zur Angabe sowohl der Planungsrichtlinie als auch der zugeordneten Agentenprioritäten zur Verfügung.

Beispiel 1: Auswahl der AIX-Planungsrichtlinie SCHED_FIFO2 mit Zuordnung einer höheren Priorität für die Prozesse zum Schreiben und Lesen des DB2-Protokolls.

```
<RESOURCE_POLICY>
 <SCHEDULING_POLICY>
  <POLICY_TYPE>SCHED_FIFO2</POLICY_TYPE>
  <PRIORITY_VALUE>60</PRIORITY_VALUE>

  <EDU_PRIORITY>
   <EDU_NAME>db2loggr</EDU_NAME>
   <PRIORITY_VALUE>56</PRIORITY_VALUE>
  </EDU_PRIORITY>

  <EDU_PRIORITY>
   <EDU_NAME>db2loggw</EDU_NAME>
   <PRIORITY_VALUE>56</PRIORITY_VALUE>
  </EDU_PRIORITY>
 </SCHEDULING_POLICY>
</RESOURCE_POLICY>
```

Beispiel 2: Ersatz für DB2NTPRICLASS=H unter Windows.

```
<RESOURCE_POLICY>
 <SCHEDULING_POLICY>
  <POLICY_TYPE>HIGH_PRIORITY_CLASS</POLICY_TYPE>
 </SCHEDULING_POLICY>
</RESOURCE_POLICY>
```

DB2_SELUDI_COMM_BUFFER

- Betriebssystem: Alle
- Standardwert: OFF, Werte: ON oder OFF
- Diese Variable wird bei der Verarbeitung von Blockcursorn über Abfragen mit SELECT from UPDATE/INSERT/DELETE (UDI) verwendet. Wenn aktiviert, verhindert diese Registrierdatenbankvariable, dass das Ergebnis einer Abfrage in einer temporären Tabelle gespeichert wird. Stattdessen versucht das DB2-Datenbanksystem bei der OPEN-Verarbeitung eines Blockcursors für eine Abfrage SELECT from UDI, das gesamte Ergebnis der Abfrage direkt im Speicherbereich des Kommunikationspuffers zu puffern.

  Anmerkung: Wenn der Kommunikationspufferbereich für das gesamte Ergebnis der Abfrage nicht ausreicht, wird der Fehler SQLCODE -906 ausgegeben und die Transaktion mit ROLLBACK rückgängig gemacht. Informationen zur Anpassung der Größe des Kommunikationspufferbereichs für lokale und ferne Anwendungen finden Sie in den Beschreibungen der Konfigurationsparameter **aslheapsz** bzw. **rqrioblk** des Datenbankmanagers.

  Diese Registrierdatenbankvariable wird bei aktivierter partitionsinterner Parallelität nicht unterstützt.
- Änderungen an dieser Variablen werden sofort wirksam und gelten für alle zukünftig kompilierten SQL-Anweisungen, wenn der Befehl **db2set** mit dem Parameter **-immediate** ausgegeben wird. Es ist nicht erforderlich, die Instanz erneut zu starten.

DB2_SET_MAX_CONTAINER_SIZE

- Betriebssystem: Alle
- Standardwert: nicht definiert, Werte: -1, beliebige positive Ganzzahl größer als 65.536 Byte
- Mit dieser Registrierdatenbankvariablen können Sie die Größe einzelner Container für Tabellenbereiche mit dynamischem Speicher bei aktivierter automatischer Größenänderungsfunktion (AutoResize) begrenzen.

  Anmerkung: Obwohl Sie **DB2_SET_MAX_CONTAINER_SIZE** in Byte, Kilobyte oder Megabyte angeben können, zeigt der Befehl **db2set** den Wert in Byte an.
- Durch den Wert -1 wird keine Begrenzung für die Größe eines Containers festgelegt.

DB2_SKIPDELETED

- Betriebssystem: Alle
- Standardwert: OFF, Werte: ON oder OFF
- Wenn diese Variable aktiviert ist, ermöglicht sie Anweisungen mit den Isolationsstufen Cursorstabilität oder Lesestabilität, gelöschte Schlüssel bei einem Indexzugriff und gelöschte Zeilen bei einem Tabellenzugriff bedingungslos zu überspringen. Wenn **DB2_EVALUNCOMMITTED** aktiviert ist, werden gelöschte Zeilen automatisch übersprungen. Nicht festgeschrie-

bene Pseudolöschungen von Schlüsseln in Indizes werden jedoch nur übersprungen, wenn auch **DB2_SKIPDELETED** aktiviert ist.

Die Variable **DB2_SKIPDELETED** ist nur anwendbar, wenn die Semantik für zurzeit festgeschriebene Daten nicht hilft, Sperrenkonflikte zu vermeiden. Wenn diese Variable definiert wird und die Semantik für zurzeit festgeschriebene Daten auf einen Suchlauf anwendbar ist, werden gelöschte Zeilen nicht übersprungen. Stattdessen wird ihre zurzeit festgeschriebene Version verarbeitet.

Diese Registrierdatenbankvariable wirkt sich nicht auf das Verhalten von Cursorn in den DB2-Katalogtabellen aus.

Diese Registrierdatenbankvariable wird mit dem Befehl **db2start** aktiviert.

DB2_SKIPINSERTED

- Betriebssystem: Alle
- Standardwert: OFF, Werte: ON oder OFF
- Wenn die Registrierdatenbankvariable **DB2_SKIPINSERTED** aktiviert ist, können Anweisungen mit der Isolationsstufe Cursorstabilität oder Lesestabilität nicht festgeschriebene eingefügte Zeilen so überspringen, als wären sie nicht eingefügt worden. Diese Registrierdatenbankvariable wirkt sich nicht auf das Verhalten von Cursorn in den DB2-Katalogtabellen aus. Diese Registrierdatenbankvariable wird beim Datenbankstart aktiviert. Die Entscheidung darüber, ob nicht festgeschriebene eingefügte Zeilen zu überspringen sind, wird beim Kompilieren oder Binden getroffen.

  Diese Registrierdatenbankvariable hat keine Auswirkung, wenn die Semantik für zurzeit festgeschriebene Daten verwendet wird. Das heißt, auch wenn **DB2_SKIPINSERTED** auf den Wert OFF gesetzt wird und das Verhalten unter der Semantik für zurzeit festgeschriebene Daten aktiviert ist, werden nicht festgeschriebene eingefügte Zeilen weiterhin übersprungen.

  Anmerkung: Das Überspringen eingefügter Zeilen ist nicht mit Tabellen kompatibel, für die eine Rolloutbereinigung ansteht. Infolgedessen ist es möglich, dass Suchoperationen auf Sperren für eine Satz-ID (RID) warten und anschließend lediglich feststellen, dass die Satz-ID zu dem durch den Rollout entfernten Block gehört.

DB2_SMS_TRUNC_TMPTABLE_THRESH

v98_u3

- Betriebssystem: Alle
- Standardwert: -2, Werte: -2, -1, 0 bis *n*, wobei *n*=Anzahl der EXTENTSIZE großen Speicherbereiche pro temporärer Tabelle im SMS-Tabellenbereichscontainer ist, die zu verwalten sind
- Diese Variable gibt einen minimalen Schwellenwert für die Dateigröße an, mit der die Datei, die eine temporäre Tabelle darstellt, in SMS-Tabellenbereichen verwaltet wird.

  Die Standardeinstellung dieser Variablen ist -2. Dies bedeutet, dass für übergelaufene temporäre SMS-Objekte, deren Größe kleiner-gleich 1 Speicherbereich (der Größe EXTENTSIZE) * Containeranzahl ist, keine unnötigen Dateisystemzugriffe stattfinden. Temporäre Objekte, die größer sind, werden auf die Größe 0 abgeschnitten.

Wird diese Variable auf 0 gesetzt, erfolgt keine spezielle Behandlung von Schwellenwerten. Stattdessen wird die Datei auf die Größe 0 abgeschnitten, wenn die Tabelle nicht mehr benötigt wird. Wenn der Wert dieser Variablen größer als 0 ist, wird eine größere Datei behalten. Objekte, deren Größe den Schwellenwert überschreitet, werden auf die Schwellenwertgröße abgeschnitten. Dadurch kann der Systemaufwand verringert werden, der mit dem Löschen und erneuten Erstellen der Datei bei jeder Verwendung einer temporären Tabelle verbunden ist.

Wenn diese Variable auf den Wert -1 gesetzt wird, wird die Datei nicht abgeschnitten und kann uneingeschränkt an Größe zunehmen, soweit die Systemressourcen dies zulassen.

DB2_SORT_AFTER_TQ

- Betriebssystem: Alle
- Standardwert: NO, Werte: YES oder NO.
- Gibt an, wie das Optimierungsprogramm mit gezielt übertragenen Tabellenwarteschlangen in einer Umgebung mit partitionierten Datenbanken arbeitet, wenn die Daten für den Empfänger sortiert sein müssen und die Anzahl der Empfängerknoten der Anzahl der Senderknoten entspricht.

  Wenn **DB2_SORT_AFTER_TQ**=NO ist, sortiert das Optimierungsprogramm Zeilen in der Regel auf der sendenden Seite und mischt die Zeilen auf der empfangenden Seite zusammen.

  Wenn **DB2_SORT_AFTER_TQ**=YES ist, überträgt das Optimierungsprogramm in der Regel die Zeilen unsortiert, mischt sie auf der empfangenden Seite nicht zusammen, sondern sortiert die Zeilen nach Empfang aller Zeilen auf der empfangenden Seite.
- Änderungen an dieser Variablen werden sofort wirksam und gelten für alle zukünftig kompilierten SQL-Anweisungen, wenn der Befehl **db2set** mit dem Parameter **-immediate** ausgegeben wird. Es ist nicht erforderlich, die Instanz erneut zu starten.

DB2_SQLWORKSPACE_CACHE

- Betriebssystem: Alle
- Standardwert: 30, Werte: 10 - 2000
- Mit dieser Variable können Sie den Umfang des Caching von zuvor genutzten Abschnitten im SQL-Arbeitsbereich steuern.

  Der SQL-Arbeitsbereich enthält in Form von Abschnitten Zuordnungen für die Ausführung von SQL. Jede SQL-Anweisung (statisch oder dynamisch), die für eine Anwendung ausgeführt wird, muss während des Zeitraums der Ausführung dieser Anweisung eine eindeutige Kopie des Abschnitts im SQL-Arbeitsbereich verwalten. Sobald die Ausführung der Anweisung abgeschlossen ist, wird der Abschnitt inaktiv und die einem inaktiven Abschnitt zugewiesenen Speicherzuordnungen können entweder freigegeben werden oder im SQL-Arbeitsbereich zwischengespeichert bleiben. Sobald eine neue Ausführung derselben SQL-Anweisung über eine beliebige Verbindung auftritt, wird möglicherweise eine zwischengespeicherte Kopie des Abschnitts im SQL-Arbeitsbereich gefunden, die von der vorherigen Ausführung stammt. Dadurch werden die Kosten für die Zuordnung und Initialisierung einer neuen Kopie des Abschnitts eingespart. Auf diese Weise enthält der SQL-Arbeitsbereich sowohl aktive Abschnitte, die aktuell ausgeführten SQL-Anweisungen entsprechen, sowie zwischengespeicherte Abschnitte, die aktuell nicht ausgeführt werden.

Der Wert dieser Registrierdatenbankvariablen gibt den Umfang der Speicherzuordnungen, deren Zwischenspeichern im SQL-Arbeitsbereich zulässig ist, in einem Prozentsatz an. Dieses Zwischenspeichern wird durch einen Prozentsatz der Speicherzuordnungen für aktive Abschnitte ausgedrückt. Daher bedeutet beispielsweise ein Wert von 50, das der SQL-Arbeitsbereich alle aktiven (zurzeit ausgeführten) Abschnitte sowie bis zu 50 Prozent der zuvor ausgeführten und nun zwischengespeicherten Abschnitte enthält, die wiederverwendet werden können. Es empfiehlt sich, die Einstellung für **DB2_SQLWORKSPACE_CACHE** anzupassen, je nachdem, welche Menge des SQL-Arbeitsbereichs für die Wiederverwendung zur Verfügung stehen soll. Wenn Sie beispielsweise den Wert dieser Variablen vergrößern, kann dies zu Leistungsverbesserungen für OLTP-Workloads führen. Andererseits bedeutet eine höhere Einstellung auch, dass die Größe des von der Anwendung gemeinsam genutzten Heapspeichers zunimmt.

Anmerkung: Wenn für den Datenbankkonfigurationsparameter **appl_memory** nicht AUTOMATIC eingestellt ist, kann die Größe des SQL-Arbeitsbereichs auch durch **appl_memory** eingeschränkt werden und der SQL-Arbeitsbereich stellt möglicherweise nicht so viel Cache zur Verfügung, wie die Einstellung für **DB2_SQLWORKSPACE_CACHE** zulassen würde. Sie sollten in einem solchen Fall erwägen, den Wert für **appl_memory** zu erhöhen (oder die Einstellung AUTOMATIC vorzunehmen).
Diese Registrierdatenbankvariable ist nicht dynamisch.

DB2_TRUSTED_BINDIN

- Betriebssystem: Alle
- Standardwert: OFF, Werte: OFF, ON oder CHECK
- Wenn die Variable **DB2_TRUSTED_BINDIN** aktiviert ist, beschleunigt sie die Ausführung von Abfrageanweisungen, die Hostvariablen innerhalb einer eingebetteten, nicht abgeschirmten gespeicherten Prozedur enthalten.

  Wenn diese Variable aktiviert ist, findet keine Konvertierung vom externen SQLDA-Format in ein internes DB2-Format beim Binden von SQL- und XQuery-Anweisungen statt, die in einer eingebetteten, nicht abgeschirmten gespeicherten Prozedur enthalten sind. Dies führt zu einer beschleunigten Verarbeitung eingebetteter SQL- und XQuery-Anweisungen.

  Die folgenden Datentypen werden in eingebetteten, nicht abgeschirmten gespeicherten Prozeduren nicht unterstützt, wenn diese Variable aktiviert ist:

  – SQL_TYP_DATE
  – SQL_TYP_TIME
  – SQL_TYP_STAMP
  – SQL_TYP_CGSTR
  – SQL_TYP_BLOB
  – SQL_TYP_CLOB
  – SQL_TYP_DBCLOB
  – SQL_TYP_CSTR
  – SQL_TYP_LSTR
  – SQL_TYP_BLOB_LOCATOR
  – SQL_TYP_CLOB_LOCATOR
  – SQL_TYP_DCLOB_LOCATOR

– SQL_TYP_BLOB_FILE
– SQL_TYP_CLOB_FILE
– SQL_TYP_DCLOB_FILE
– SQL_TYP_BLOB_FILE_OBSOLETE
– SQL_TYP_CLOB_FILE_OBSOLETE
– SQL_TYP_DCLOB_FILE_OBSOLETE

Wenn diese Datentypen auftreten, wird ein Fehler mit dem SQLCODE-Wert -804 und dem SQLSTATE-Wert 07002 zurückgegeben.

Anmerkung: Der Datentyp und die Länge der Eingabehostvariablen muss exakt mit dem internen Datentyp und der Länge des entsprechenden Elements übereinstimmen. Für Hostvariablen wird diese Voraussetzung immer erfüllt. Bei Parametermarken muss jedoch sorgfältig darauf geachtet werden, dass übereinstimmende Datentypen verwendet werden. Zur Sicherstellung, dass die Datentypen und Längen für alle Eingabehostvariablen übereinstimmen, kann die Option `CHECK` verwendet werden. Allerdings macht diese Option die meisten Leistungsvorteile wieder zunichte.

Anmerkung: **`DB2_TRUSTED_BINDIN`** ist veraltet und wird in einem späteren Release entfernt.

`DB2_USE_ALTERNATE_PAGE_CLEANING`

- Betriebssystem: Alle
- Standardwert: nicht definiert, Werte: `ON` oder `OFF`
- Diese Variable gibt an, ob eine DB2-Datenbank die alternative Methode der Seitenlöschalgorithmen oder die Standardmethode der Seitenlöschfunktion verwendet. Wenn diese Variable auf den Wert `ON` gesetzt ist, verwendet das DB2-System eine Seitenlöschmethode, die geänderte Seiten auf die Platte schreibt, dem Wert von LSN_GAP voraus bleibt und proaktiv Seiten ermittelt, die bereinigt werden können. Dadurch können die Seitenlöschfunktionen die verfügbare E/A-Bandbreite der Platte besser nutzen. Wenn diese Variable auf den Wert `ON` gesetzt ist, ist der Datenbankkonfigurationsparameter *chngpgs_thresh* nicht länger relevant, weil er keinen Einfluss auf die Aktivitäten der Seitenlöschfunktionen hat.

`DB2_USE_FAST_PREALLOCATION`

- Betriebssystem: AIX, Linux und Solaris mit den Dateisystemen Veritas-VxFS, JFS2, GPFS sowie ext4 (nur Linux)
- Standardwert: `ON` für Veritas VxFS, JFS2, GPFS und ext4, Werte: `ON` oder `OFF`
- Diese Variable ermöglicht es der Dateisystemfunktion für die schnelle Vorabzuordnung, einen Tabellenbereich zu reservieren und das Erstellen oder Ändern großer Tabellenbereiche sowie das Ausführen von Datenbankrestoreoperationen zu beschleunigen. Für die Implementierung dieser Geschwindigkeitsverbesserung fällt nur ein geringer Deltaaufwand an, wenn während der Laufzeit die eigentliche Bereichszuordnung beim Einfügen von Zeilen durchgeführt wird.

  Zum Inaktivieren der schnellen Vorabzuordnung setzen Sie **`DB2_USE_FAST_PREALLOCATION`** auf OFF. Hierdurch kann die Laufzeitleistung verbessert werden, die Tabellenbereichserstellung und der Datenbankrestore werden jedoch verlangsamt. Dies gilt für eine Reihe von Betriebssystemen, vor allem für AIX, wenn ein großes Volumen an Einfüge- und Auswahloperationen für denselben Tabellenbereich anfällt. Beachten

Sie, dass nach dem Inaktivieren der schnellen Vorabzuordnung ein Restore der Datenbank durchgeführt werden muss.

DB2_USE_IOCP

- Betriebssystem: AIX
- Standardwert: ON, Werte: ON oder OFF
- Diese Variable ermöglicht die Verwendung von AIX-E/A-Ausführungsports (IOCP - I/O Completion Ports) bei der Übergabe und Erfassung asynchroner E/A-Anforderungen (AIO-Anforderungen). Diese Funktion dient zur Erhöhung der Leistung in einer NUMA-Umgebung (NUMA - Non-Uniform Memory Access, nicht gleichmäßiger Speicherzugriff), indem der ferne Speicherzugriff vermieden wird.

Verschiedene Variablen

DB2ADMINSERVER

- Betriebssystem: Windows und UNIX
- Standardwert: NULL
- Gibt den DB2-Verwaltungsserver (DAS, DB2 Administration Server) an.

DB2_ATS_ENABLE

- Betriebssystem: Alle
- Standardwert: NULL, Werte: YES/TRUE/ON/1 oder NO/FALSE/OFF/0
- Diese Variable steuert, ob der Scheduler für Verwaltungstasks (ATS - Administrative Task Scheduler) ausgeführt wird. Der Scheduler für Verwaltungstasks ist standardmäßig inaktiviert. Wenn der Scheduler inaktiviert ist, können Sie die integrierten Prozeduren und Sichten zum Definieren und Ändern von Tasks verwenden, jedoch führt der Scheduler die Tasks nicht aus.
- Änderungen an dieser Variable werden sofort wirksam und gelten für alle zukünftigen kompilierten SQL-Anweisungen. Es ist nicht nötig, die Instanz erneut zu starten oder den Befehl **db2set** mit dem Parameter **-immediate** abzusetzen.

DB2AUTH

- Betriebssystem: Alle
- Standardwert: nicht definiert. Werte: DISABLE_CHGPASS, OSAUTHDB, SQLADM-_NO_RUNSTATS_REORG, TRUSTEDCLIENT_DATAENC, TRUSTEDCLIENT_SRVRENC
- Mit dieser Variablen können Sie die Funktionsweise der Benutzerauthentifizierung optimieren. Gültige Werte:
  - DISABLE_CHGPASS: Durch diesen Wert wird die Möglichkeit, das Kennwort vom Client aus zu ändern, inaktiviert.
  - OSAUTHDB: Dieser Wert weist den DB2-Datenbankmanager an, die Authentifizierungs- und Gruppeneinstellung für einen Benutzer im Betriebssystem AIX zu verwenden. Die transparente LDAP-Unterstützung wurde auch auf die Betriebssysteme Linux, HP-UX und Solaris erweitert. Bei dem LDAP-Server kann es sich um einen beliebigen der folgenden handeln:
    - IBM Tivoli Directory Server (ITDS)
    - Microsoft Active Directory (MSAD)
    - Sun One Directory Server

– SQLADM_NO_RUNSTATS_REORG: Dieser in DB2 Version 9.7 Fixpack 5 eingeführte Wert inaktiviert für Benutzer mit der Berechtigung SQLADM die Möglichkeit, eine REORG- oder RUNSTATS-Operation auszuführen.
– TRUSTEDCLIENT_DATAENC: Durch diesen Wert werden ungesicherte Clients dazu gezwungen, DATA_ENCRYPT zu verwenden. Dieser Wert kann nicht auf einen DB2 Connect-Gateway angewendet werden.
– TRUSTEDCLIENT_SRVRENC: Durch diesen Wert werden ungesicherte Clients dazu gezwungen, SERVER_ENCRYPT zu verwenden. Dieser Wert kann nicht auf einen DB2 Connect-Gateway angewendet werden.

DB2CLIINIPATH

- Betriebssystem: Alle
- Standardwert: NULL
- Dient zum Überschreiben des Standardpfads der CLI/ODBC-Konfigurationsdatei (db2cli.ini) und zum Angeben einer alternativen Speicherposition auf dem Client. Der definierte Wert muss ein gültiger Pfad auf dem Clientsystem sein.

DB2_COMMIT_ON_EXIT

- Betriebssystem: UNIX
- Standardwert: OFF, Werte: OFF/NO/0 oder ON/YES/1
- Unter UNIX-Betriebssystemen vor DB2 UDB Version 8 schrieb DB2 alle verbleibenden unvollständigen Transaktionen bei erfolgreicher Beendigung der Anweisung fest. In DB2 UDB Version 8 wurde diese Funktionsweise geändert, sodass unvollständige Transaktionen bei Beendigung mit ROLLBACK rückgängig gemacht wurden. Diese Registrierdatenbankvariable gibt Benutzern von Anwendungen mit eingebettetem SQL, die von der früheren Funktionsweise abhängig sind, die Möglichkeit, diese Funktionsweise in DB2 Version 9 weiterhin zu aktivieren. Diese Registrierdatenbankvariable hat keine Auswirkungen auf JDBC-, CLI- und ODBC-Anwendungen.

  Beachten Sie, dass diese Registrierdatenbankvariable veraltet ist und dass die Funktionsweise mit Commit bei Beendigung in zukünftigen Releases nicht weiter unterstützt wird. Benutzer sollten bestimmen, ob ihre Anwendungen, die vor DB2 Version 9 entwickelt wurden, weiterhin von dieser Funktionsweise abhängig sind, und die entsprechenden expliziten COMMIT- bzw. ROLLBACK-Anweisungen nach Bedarf in die Anwendungen einfügen. Wenn die Registrierdatenbankvariable aktiviert ist, sollte sorgsam vermieden werden, neue Anwendungen zu implementieren, die keine expliziten COMMIT-Anweisungen vor ihrer Beendigung ausführen.

  Die meisten Benutzer sollten die Standardeinstellung dieser Registrierdatenbankvariablen belassen.

DB2_COMMON_APP_DATA_PATH

- Betriebssystem: Windows
- Standardwert: Pfad zu allgemeinen Windows-Anwendungsdaten.
  – Für Windows XP- und Windows 2003-Betriebssysteme: C:\Documents and Settings\All Users\Application Data\
  – Für Windows Vista-Betriebssysteme und spätere Betriebssysteme: C:\Programme\
- Verweist auf die benutzerdefinierte Position, an der sich die gemeinsamen DB2-Anwendungsdaten für die DB2-Kopie befinden. Diese Regist-

rierdatenbankvariable wird gefüllt, wenn **DB2_COMMON_APP_DATA_TOP_PATH** während der Installation mit der Antwortdatei angegeben wird oder wenn das Feld "DB2 Common Application Data Top Path" während der angepassten Installation gefüllt wird.

Ab V9.7 Fix Pack 5 ist diese Registrierdatenbankvariable im Befehl **db2set** sichtbar, kann aber nicht geändert werden. Jeder Versuch, einen gegebenen Registrierungswert zu ändern, führt zu Fehler DBI1301E, der auf ungültige Werte hinweist.

DB2_COMPATIBILITY_VECTOR

- Betriebssystem: Alle
- Standardwert: NULL, Werte: NULL oder 00 bis FFF
- Die Registrierdatenbankvariable **DB2_COMPATIBILITY_VECTOR** dient zum Aktivieren einer oder mehrerer DB2-Kompatibilitätsfunktionen, die seit DB2 Version 9.5 eingeführt wurden. Diese Funktionen vereinfachen die Aufgabe der Migration von Anwendungen, die für andere Anbieter von relationalen Datenbanken geschrieben wurden, auf DB2 Version 9.5 oder spätere Versionen.
- Die Variable **DB2_COMPATIBILITY_VECTOR** enthält einen Hexadezimalwert. Jedes Bit des Variablenwerts aktiviert eine der DB2-Kompatibilitätsfunktionen, wie in der Tabelle DB2_COMPATIBILITY_VECTOR-Werte aufgeführt. Zur Aktivierung aller unterstützten Kompatibilitätsfunktionen setzen Sie die Registrierdatenbankvariable auf den Wert ORA (der mit dem Hexadezimalwert FFF äquivalent ist). Dies ist die empfohlene Einstellung.

DB2CONNECT_DISCONNECT_ON_INTERRUPT

- Betriebssystem: Alle
- Standardwert: NO, Werte: YES/TRUE/1 oder NO/FALSE/0
- Wenn diese Variable auf den Wert YES (TRUE oder 1) gesetzt ist, gibt sie an, dass die Verbindung zu einem z/OS-Server mit DB2 Universal Database der Version 8 (oder einer höheren Version) sofort abzubrechen ist, wenn ein Interrupt auftritt. Sie können diese Variable in den folgenden Konfigurationen verwenden:
  - Wenn Sie einen DB2-Client mit einem z/OS-Server mit DB2 UDB der Version 8 (oder einer höheren Version) ausführen, setzen Sie die Variable **DB2CONNECT_DISCONNECT_ON_INTERRUPT** auf dem Client auf den Wert YES.
  - Wenn Sie einen DB2-Client über ein DB2 Connect-Gateway mit einem z/OS-Server mit DB2 UDB der Version 8 (oder einer höheren Version) ausführen, setzen Sie die Variable **DB2CONNECT_DISCONNECT_ON_INTERRUPT** auf dem Gateway auf den Wert YES.

DB2_CREATE_DB_ON_PATH

- Betriebssystem: Windows
- Standardwert: NULL, Werte: YES oder NO
- Setzen Sie diese Registrierdatenbankvariable auf den Wert YES, um die Unterstützung für die Verwendung eines Pfads (und eines Laufwerks) als Datenbankpfad zu aktivieren. Die Einstellung der Variablen **DB2_CREATE_DB_ON_PATH** wird überprüft, wenn eine Datenbank erstellt wird, wenn der Konfigurationsparameter **dftdbpath** des Datenbankma-

nagers definiert wird und wenn ein Restore einer Datenbank durchgeführt wird. Der vollständig qualifizierte Datenbankpfad kann bis zu 215 Zeichen lang sein.

Wenn **DB2_CREATE_DB_ON_PATH** nicht definiert wird (oder auf den Wert NO gesetzt wird) und Sie beim Erstellen oder beim Restore einer Datenbank einen Pfad für den Datenbankpfad angeben, wird der Fehler SQL1052N zurückgegeben.

Wenn **DB2_CREATE_DB_ON_PATH** nicht definiert wird (oder auf den Wert NO gesetzt wird) und Sie den Konfigurationsparameter **dftdbpath** der Datenbank aktualisieren, wird der Fehler SQL5136N zurückgegeben.

Vorsicht:
Wenn die Pfadunterstützung zur Erstellung neuer Datenbanken verwendet wird, ist es möglich, dass vor DB2 Version 9.1 geschriebene Anwendungen, die die API db2DbDirGetNextEntry() oder eine ältere Version dieser API verwenden, nicht korrekt funktionieren. Detaillierte Informationen zu verschiedenen Szenarios sowie zur geeigneten Vorgehensweise finden Sie unter http://www.ibm.com/software/data/db2/support/db2_9/.

DB2_DDL_SOFT_INVAL

- Betriebssystem: Alle
- Standardwert: ON, Werte: ON oder OFF
- Ermöglicht eine vorläufige Inaktivierung (Ungültigmachung) betroffener Datenbankobjekte, wenn sie gelöscht oder geändert werden.

  Wenn die Variable **DB2_DDL_SOFT_INVAL** auf den Wert ON gesetzt ist, können beliebige DDL-Operationen (z. B. DROP, ALTER oder DETACH) starten, ohne darauf zu warten, dass Transaktionen, die auf dieselben Objekte verweisen, abgeschlossen werden. Aktuelle Ausführungen, die von den Objekten abhängig sind, verwenden weiterhin die ursprüngliche Objektdefinition, während neue Ausführungen das geänderte Objekt verwenden. Dies ermöglicht einen besseren gemeinsamen Zugriff während der Ausführung von DDL-Anweisungen (DDL, Data Definition Language).

  Anmerkung: Die neuen Möglichkeiten für vorläufige Inaktivierung gelten nur für dynamische Pakete. Objekte mit statischen Paketen erfordern weiterhin eine absolute Inaktivierung.

DB2_DISABLE_FLUSH_LOG

- Betriebssystem: Alle
- Standardwert: OFF, Werte: ON oder OFF
- Gibt an, ob das Schließen der aktiven Protokolldatei nach Abschluss eines Online-Backups inaktiviert werden soll.

  Wenn ein Online-Backup abgeschlossen ist, wird die letzte aktive Protokolldatei abgeschnitten, geschlossen und zur Archivierung verfügbar gemacht. Hierdurch wird sichergestellt, dass Ihrem Online-Backup alle zur Recovery benötigten archivierten Protokolldateien zur Verfügung stehen. Sie möchten möglicherweise das Schließen der letzten aktiven Protokolldatei inaktivieren, wenn Sie Sorge haben, dass Sie Teile des Protokollfolgenummernbereichs (LSN) verschwenden. Jedes Mal, wenn eine aktive Protokolldatei abgeschnitten wird, wird die Protokollfolgenummer um einem zum abgeschnittenen Speicherplatz proportionalen Betrag er-

höht. Wenn Sie jeden Tag eine große Anzahl von Online-Backups durchführen, ist es vielleicht sinnvoll, das Schließen der letzten aktiven Protokolldatei zu inaktivieren.

Darüber hinaus kann es sinnvoll sein, das Schließen der letzten aktiven Protokolldatei zu inaktivieren, wenn Sie feststellen, dass Sie Nachrichten über ein volles Protokoll kurz nach Abschluss des Online-Backups empfangen. Wenn eine Protokolldatei abgeschnitten wird, wird der reservierte Speicherbereich für aktive Protokolle um einen Betrag vergrößert, der proportional zur Größe des abgeschnittenen Protokolls ist. Der Speicherbereich für aktive Protokolle wird freigegeben, wenn die abgeschnittene Protokolldatei wieder zur Verwendung verfügbar gemacht wird. Die Wiederverfügbarmachung erfolgt kurz nach dem Zeitpunkt, zu dem die Protokolldatei inaktiv wird. Während des kurzen Intervalls zwischen diesen beiden Ereignissen empfangen Sie möglicherweise Nachrichten über ein volles Protokoll.

Bei jedem Backup, das Protokolle mit einbezieht, wird diese Registrierdatenbankvariable ignoriert, da die aktive Protokolldatei abgeschnitten und geschlossen werden muss, um die Protokolle mit in das Backup einzubeziehen.

- Änderungen an dieser Variable werden sofort wirksam und gelten für alle zukünftigen kompilierten SQL-Anweisungen. Es ist nicht nötig, die Instanz erneut zu starten oder den Befehl **db2set** mit dem Parameter **-immediate** abzusetzen.

DB2_DISPATCHER_PEEKTIMEOUT

- Betriebssystem: Alle
- Standardwert: 1, Werte: 0 bis 32767 Sekunden; 0 gibt an, dass das Zeitlimit sofort abläuft
- Mit **DB2_DISPATCHER_PEEKTIMEOUT** können Sie die Zeit (in Sekunden) anpassen, die eine Zuteilerroutine (Dispatcher) auf die Verbindungsanforderung eines Clients wartet, bevor sie den Client an einen Agenten weitergibt. In den meisten Fällen sollte eine Anpassung dieser Registrierdatenbankvariablen nicht nötig sein. Diese Registrierdatenbankvariable wirkt sich nur auf Instanzen aus, für die der DB2 Connect-Verbindungskonzentrator aktiviert ist.

  Diese Registrierdatenbankvariable und die Registrierdatenbankvariable **DB2_SERVER_CONTIMEOUT** dienen beide zur Konfiguration der Behandlung eines neuen Clients während der Verbindungszeit. Wenn viele langsame Clients eine Verbindung zu einer Instanz herstellen, wird die Zuteilerroutine möglicherweise bis zu 1 Sekunde aufgehalten, um das Zeitlimit jedes Clients abzuwarten. Auf diese Weise wird die Zuteilerroutine zu einem Engpass, wenn viele Clients gleichzeitig eine Verbindung herstellen. Wenn in einer Instanz mit mehreren aktiven Datenbanken sehr langsame Verbindungszeiten festgestellt werden, kann **DB2_DISPATCHER_PEEKTIMEOUT** auf den Wert 0 herabgesetzt werden. Die Herabsetzung von **DB2_DISPATCHER_PEEKTIMEOUT** bewirkt, dass die Zuteilerroutine nur kurz prüft, ob die Verbindungsanforderung des Clients bereits da ist, ohne auf das Eintreffen der Verbindungsanforderung zu warten. Wenn ein ungültiger Wert definiert wird, wird der Standardwert verwendet. Diese Registrierdatenbankvariable ist nicht dynamisch.

DB2_DJ_INI

- Betriebssystem: Alle
- Standardwert:

- UNIX: *db2-instanzverzeichnis*/cfg/db2dj.ini
- Windows: *db2-installationsverzeichnis*\cfg\db2dj.ini

• Gibt den absoluten Pfadnamen der Konfigurationsdatei für die Föderation von Datenquellen an, zum Beispiel: db2set DB2_DJ_INI=$HOME/sqllib/cfg/my_db2dj.ini. Diese Datei enthält die Einstellungen für Umgebungsvariablen von Datenquellen. Diese Umgebungsvariablen werden vom Informix-Wrapper und von den durch InfoSphere Federation Server bereitgestellten Wrappern verwendet.

Das folgende Beispiel zeigt eine Konfigurationsdatei für die Föderation von Datenquellen:

```
INFORMIXDIR=/informix/client_sdk
INFORMIXSERVER=inf93
ORACLE_HOME=/usr/oracle9i
SYBASE=/sybase/V12
SYBASE_OCS=OCS-12_5
```

Für die Datei db2dj.ini gelten die folgenden Einschränkungen:

- Einträge müssen das Format *umgebvarname=wert* haben, wobei *umgebvarname* der Name der Umgebungsvariablen und *wert* der entsprechende Wert ist.
- Der Name der Umgebungsvariablen kann maximal 255 Byte lang sein.
- Der Wert der Umgebungsvariablen kann maximal 765 Byte lang sein.

Diese Variable wird ignoriert, sofern nicht der Parameter **federated** des Datenbankmanagers den Wert YES hat.

DB2_DMU_DEFAULT

• Betriebssystem: Alle

• Standardwert: NULL, Werte: IMPLICITLYHIDDENMISSING, IMPLICITLYHIDDENINCLUDE

• Mit dieser Variable können Sie das Standardverhalten festlegen, um implizit verdeckte Spalten ein- bzw. auszuschließen, wenn die Spaltenliste von den Dienstprogrammen LOAD, EXPORT, dem Importdienstprogramm oder dem Aufnahmedienstprogramm übergangen wird. Gültige Werte:

NULL
Dies bedeutet, dass kein Standardverhalten angegeben ist. Wenn die Tabelle implizit verdeckte Spalten enthält, muss die Spaltenliste explizit angegeben werden, andernfalls müssen von den Dienstprogrammen die Optionen für verdeckte Spalten angegeben werden. Andernfalls tritt ein Fehler auf.

IMPLICITLYHIDDENMISSING
Die Dienstprogramme setzen voraus, dass die implizit verdeckten Spalten standardmäßig nicht eingeschlossen sind, es sei denn, die Spaltenliste oder die Optionen für verdeckte Spalten sind angegeben.

IMPLICITLYHIDDENINCLUDE
Die Dienstprogramme setzen voraus, dass die implizit verdeckten Spalten standardmäßig eingeschlossen sind, wenn weder die Spaltenliste noch die Optionen für verdeckte Spalten angegeben sind.

Anhand der folgenden Beispiele können Sie erkennen, welche Auswirkungen die Einstellung für **DB2_DMU_DEFAULT** auf das Ergebnis einer Ladeoperation hat:

– Für **DB2_DMU_DEFAULT** wird IMPLICITLYHIDDENMISSING festgelegt.

```
db2 load from delfile1 of del insert into table1
```

Wenn 'table1' implizit verdeckte Spalten aufweist, setzt das Dienstprogramm LOAD voraus, dass sich die Daten für implizit verdeckte Spalten nicht in der Eingabedatei befinden.

– Für **DB2_DMU_DEFAULT** wird IMPLICITLYHIDDENINCLUDE festgelegt.

```
db2 load from delfile1 of del insert into table1
```

Wenn 'table1' implizit verdeckte Spalten aufweist, setzt das Dienstprogramm LOAD voraus, dass sich die Daten für implizit verdeckte Spalten in der Eingabedatei befinden und versucht, diese zu laden.

DB2_DOCHOST

- Betriebssystem: Alle
- Standardwert: nicht definiert (DB2 versucht dennoch, über die IBM Website auf das Information Center zuzugreifen); Werte: http://*hostname*, wobei *hostname* ein gültiger Hostname oder eine gültige IP-Adresse ist
- Gibt den Namen des Hosts an, auf dem das *DB2 Information Center* installiert ist. Diese Variable kann bei der Installation des *DB2 Information Center* automatisch definiert werden, wenn die Option zur automatischen Konfiguration im DB2-Installationsassistenten ausgewählt wird.
- Änderungen an dieser Variable werden sofort wirksam und gelten für alle zukünftigen kompilierten SQL-Anweisungen. Es ist nicht nötig, die Instanz erneut zu starten oder den Befehl **db2set** mit dem Parameter **-immediate** abzusetzen.

DB2_DOCPORT

- Betriebssystem: Alle
- Standardwert: NULL, Werte: jede gültige Portnummer
- Gibt die Portnummer an, über die das DB2-Hilfesystem die DB2-Dokumentation bereitstellt. Diese Variable kann bei der Installation des *DB2 Information Center* automatisch definiert werden, wenn die Option zur automatischen Konfiguration im DB2-Installationsassistenten ausgewählt wird.
- Änderungen an dieser Variable werden sofort wirksam und gelten für alle zukünftigen kompilierten SQL-Anweisungen. Es ist nicht nötig, die Instanz erneut zu starten oder den Befehl **db2set** mit dem Parameter **-immediate** abzusetzen.

DB2DSDRIVER_CFG_PATH

- Betriebssystem: Alle
- Standardwert: NULL
- Dient zum Überschreiben des Standardpfads der Konfigurationsdatei db2dsdriver.cfg. Weitere Informationen finden Sie im Abschnitt 'Zugehörige Referenzen'.

DB2DSDRIVER_CLIENT_HOSTNAME

- Betriebssystem: Alle
- Standardwert: NULL
- Dient zum Überschreiben des Standardnamens für den Client-Host der Konfigurationsdatei (db2dsdriver.cfg). Mit dieser Variable wird erzwun-

gen, dass CLI den Namenseintrag für den Client-Host aus dem Abschnitt für die automatische Clientweiterleitung in der Datei db2dsdriver.cfg abruft.

DB2_ENABLE_AUTOCONFIG_DEFAULT

- Betriebssystem: Alle
- Standardwert: NULL, Werte: YES oder NO
- Mit dieser Variable wird gesteuert, ob der Konfigurationsadvisor automatisch bei der Datenbankerstellung ausgeführt wird. Wenn die Variable **DB2_ENABLE_AUTOCONFIG_DEFAULT** nicht definiert (null) ist, bewirkt dies dasselbe wie die Einstellung der Variablen auf den Wert YES. Das heißt, der Konfigurationsadvisor wird bei der Datenbankerstellung ausgeführt. Sie brauchen die Instanz nach dem Definieren dieser Variablen nicht erneut zu starten. Wenn Sie den Befehl **AUTOCONFIGURE** oder die Anweisung **CREATE DB AUTOCONFIGURE** ausführen, überschreiben diese die Einstellung von **DB2_ENABLE_AUTOCONFIG_DEFAULT**.
- Änderungen an dieser Variable werden sofort wirksam und gelten für alle zukünftigen kompilierten SQL-Anweisungen. Es ist nicht nötig, die Instanz erneut zu starten oder den Befehl **db2set** mit dem Parameter **-immediate** abzusetzen.

DB2_ENABLE_LDAP

- Betriebssystem: Alle
- Standardwert: NO, Werte: YES oder NO.
- Gibt an, ob LDAP (Lightweight Directory Access Protocol) verwendet wird. LDAP ist eine Zugriffsmethode auf Verzeichnisservices.

DB2_EVMON_EVENT_LIST_SIZE

- Betriebssystem: Alle
- Standardwert: 0 (keine Begrenzung), Werte: Ein in KB (Kb/kb), MB (Mb/mb) oder GB (Gb/gb) angegebener Wert. Obwohl es keine feste obere Grenze für diese Variable gibt, wird sie durch die Größe des verfügbaren Speichers aus dem Monitorzwischenspeicher begrenzt.
- Diese Registrierdatenbankvariable gibt die maximale Anzahl Byte an, die in einer Warteschlange gesammelt werden können, indem sie darauf warten, in einen bestimmten Ereignismonitor geschrieben zu werten. Wenn diese Begrenzung erreicht wird, warten Agenten, die versuchen, Ereignismonitorsätze zu senden, bis die Größe der Warteschlange unter diesen Schwellenwert sinkt.
- Änderungen an dieser Variable werden sofort wirksam und gelten für alle zukünftigen kompilierten SQL-Anweisungen. Es ist nicht nötig, die Instanz erneut zu starten oder den Befehl **db2set** mit dem Parameter **-immediate** abzusetzen.

Anmerkung: Wenn Aktivitätssätze nicht aus dem Monitorzwischenspeicher zugeordnet werden können, werden sie übergangen. Um dies zu vermeiden, setzen Sie den Konfigurationsparameter **mon_heap_sz** auf den Wert AUTOMATIC. Wenn der Konfigurationsparameter **mon_heap_sz** auf einen bestimmten Wert gesetzt wird, stellen Sie sicher, dass die Variable **DB2_EVMON_EVENT_LIST_SIZE** auf einen kleineren Wert gesetzt wird. Diese Maßnahmen können jedoch nicht garantieren, dass Aktivitätssätze nicht übergangen werden, da der Monitorzwischenspeicher auch zur Verfolgung anderer Monitorelemente verwendet wird.

DB2_EVMON_STMT_FILTER

- Betriebssystem: Alle
- Standardwert: nicht definiert, Werte:
  - `ALL`: Gibt an, dass die Ausgabe für alle Anweisungsereignismonitore zu filtern ist. Diese Option ist exklusiv.
  - '*nameA nameB nameC*': Jeder Name in der Zeichenfolge stellt den Namen eines Ereignismonitors dar, für den Datensätze zu filtern sind. Wenn mehr als ein Name angegeben wird, muss jeder Name durch ein und nur ein Leerzeichen getrennt angegeben werden. Alle Eingabenamen werden von DB2 in Großbuchstaben umgesetzt. Die maximale Anzahl von Ereignismonitoren, die Sie angeben können, beträgt 32. Jeder Monitorname kann maximal bis zu 18 Zeichen lang sein.
  - '*nameA:op1,op2 nameB:op1,op2 nameC:op1*': Jeder Name in der Zeichenfolge stellt den Namen eines Ereignismonitors dar, für den Datensätze zu filtern sind. Jede Option (*op1*, *op2* usw.) stellt einen ganzzahligen Wert dar, der einer bestimmten SQL-Operation zugeordnet ist. Die Angabe ganzzahliger Werte ermöglicht Benutzern festzustellen, welche Regeln auf welche Ereignismonitore anzuwenden sind.
- Die Variable **DB2_EVMON_STMT_FILTER** kann dazu verwendet werden, die Anzahl von Datensätzen zu verringern, die durch einen Anweisungsereignismonitor geschrieben werden. Wenn sie definiert wird, sorgt diese Registrierdatenbankvariable dafür, dass nur die Datensätze für die folgenden SQL-Operationen an den angegebenen Ereignismonitor geschrieben werden:

*Tabelle 124. Werte, die für **DB2_EVMON_STMT_FILTER** zur Begrenzung der Ereignismonitorausgabe auf bestimmte SQL-Operationen zu verwenden sind*

SQL-Operation	Zuordnung ganzzahliger Werte
SELECT	15
EXECUTE	2
EXECUTE_IMMEDIATE	3
CLOSE	6
STATIC COMMIT	8
STATIC ROLLBACK	9
CALL	12
PRE_EXEC	17

Alle anderen Operationen werden in der Ausgabe des Anweisungsereignismonitors nicht berücksichtigt. Um die Gruppe von Operationen anzupassen, für die Datensätze in den Ereignismonitor geschrieben werden, verwenden Sie ganzzahlige Werte.

Beispiel 1:

```
db2set DB2_EVMON_STMT_FILTER= 'mon1 monitor3'
```

In diesem Beispiel empfangen die Ereignismonitore 'mon1' und 'monitor3' einen Datensatz für eine eingeschränkte Liste von Anwendungsanforderungen. Wenn zum Beispiel eine Anwendung, die durch den Anweisungsereignismonitor 'mon1' überwacht wird, eine dynamische SQL-Anweisung vorbereitet, auf der Basis dieser Anweisung einen Cursor öffnet, 10.000 Zeilen aus diesem Cursor abruft und anschließend eine Anforderung zum Schließen des Cursors absetzt, wird nur ein Datensatz für die Schließanforderung in der Ausgabe des Ereignismonitors 'mon1' angezeigt.

Beispiel 2:

```
db2set DB2_EVMON_STMT_FILTER='evmon1:3,8 evmon2:9,15
```

In diesem Beispiel empfangen 'evmon1' und 'evmon2' einen Datensatz für eine eingeschränkte Liste von Anwendungsanforderungen. Wenn z. B. eine Anwendung, die durch den Anweisungsereignismonitor 'evmon1' überwacht wird, eine Anweisung CREATE absetzt, werden nur die Operationen EXECUTE IMMEDIATE und STATIC COMMIT in der Ausgabe des Ereignismonitors 'evmon1' angezeigt. Wenn eine Anwendung, die durch den Anweisungsereignismonitor 'evmon2' überwacht wird, SQL mit SELECT und STATIC ROLLBACK ausführt, werden nur diese zwei Operationen in der Ausgabe des Ereignismonitors 'evmon2' angezeigt.

Anmerkung: Definitionen von Konstanten für Datenbanksystemmonitore finden Sie in der Headerdatei `sqlmon.h`.

DB2_EXTSECURITY

- Betriebssystem: Windows
- Standardwert: `YES`, Werte: `YES` oder `NO`.
- Verhindert den unbefugten Zugriff auf DB2 durch Sperren der DB2-Objekte (Systemdateien, Verzeichnisse und IPC-Objekte). Zur Vermeidung potenzieller Probleme sollte diese Registrierdatenbankvariable nicht inaktiviert werden. Wenn **DB2_EXTSECURITY** nicht definiert wird, wird der Wert `YES` für DB2-Datenbankserverprodukte und `NO` für Clients angenommen.

DB2_FALLBACK

- Betriebssystem: Windows
- Standardwert: `OFF`, Werte: `ON` oder `OFF`
- Mit dieser Variablen können Sie alle Datenbankverbindungen bei der Verarbeitung einer Datenbankrückübertragung (Fallback) zwangsweise trennen. Sie wird in Verbindung mit der Unterstützung für die Funktionsübernahme (Failover) in der Windows-Umgebung mit Microsoft Cluster Server (MSCS) verwendet. Wenn **DB2_FALLBACK** nicht definiert oder auf `OFF` gesetzt ist und bei der Rückübertragung (Fallback) eine Datenbankverbindung besteht, kann die DB2-Ressource nicht in den Offlinestatus versetzt werden. Dies bedeutet, dass die Rückübertragung fehlschlägt.

DB2_FMP_COMM_HEAPSZ

- Betriebssystem: Windows, UNIX
- Standardwert: 20 MB oder ausreichend Speicherbereich, um 10 abgeschirmte Routinen ausführen zu können (je nachdem, welcher Wert größer ist). Unter AIX ist der Standardwert 256 MB.
- Diese Variable gibt die Größe des Pools (in 4-KB-Seiten) an, der für Aufrufe abgeschirmter Routinen, wie zum Beispiel Aufrufe gespeicherter Prozeduren oder benutzerdefinierter Funktionen, verwendet wird. Der von den einzelnen abgeschirmten Routinen verwendete Speicherbereich entspricht dem zweifachen Wert des Konfigurationsparameters **aslheapsz**.

  Wenn Sie auf Ihrem System eine große Anzahl an abgeschirmten Routinen ausführen, müssen Sie den Wert dieser Variablen unter Umständen erhöhen. Wenn Sie eine sehr kleine Anzahl an abgeschirmten Routinen ausführen, können Sie den Wert senken.

Wenn dieser Wert auf 0 gesetzt wird, bedeutet dies, dass keine Gruppe erstellt wird, sodass keine abgeschirmten Routinen aufgerufen werden können. Dies bedeutet außerdem, dass der Diagnosemonitor und die Funktionalität zur automatischen Datenbankpflege (z. B. automatische Backups, Statistikerfassungen und **REORG**) inaktiviert wird, da diese Funktionalität von der Infrastruktur für abgeschirmte Routinen abhängig ist.

DB2_GRP_LOOKUP

- Betriebssystem: Windows
- Standardwert: NULL, Werte: LOCAL, DOMAIN, TOKEN, TOKENLOCAL, TOKENDOMAIN
- Diese Variable gibt an, welcher Windows-Sicherheitsmechanismus zur Aufzählung der Gruppen verwendet wird, denen ein Benutzer angehört.

DB2_HADR_BUF_SIZE

- Betriebssystem: Alle
- Standardwert: 2***logbufsz***
- Diese Variable gibt die Protokollempfangspuffergröße des Bereitschaftssystems in Einheiten von Protokollseiten an. Wenn nicht definiert, verwendet DB2 das Zweifache des Werts des Konfigurationsparameters **logbufsz** für die Empfangspuffergröße des Bereitschaftssystems. Die maximale Größe, die angegeben werden kann, ist 4 GB. Diese Variable muss in der Bereitschaftsinstanz definiert werden. Sie wird von der Primärdatenbank ignoriert.

  Wenn der HADR-Synchronisationsmodus (Datenbankkonfigurationsparameter **hadr_syncmode**) auf den Wert ASYNC gesetzt ist, kann ein langsames Bereitschaftssystem im Peerstatus die Sendeoperation auf dem Primärsystem aufhalten und so die Transaktionsverarbeitung in der Primärdatenbank blockieren. In einer Bereitschaftsdatenbank kann ein Protokollempfangspuffer mit einer größeren als der Standardgröße konfiguriert werden, um die Aufnahme einer größeren Menge unverarbeiteter Protokolldaten zu ermöglichen. Dadurch lassen sich kurze Zeiträume auffangen, in denen die Primärdatenbank schneller Protokolldaten generiert, als die Bereitschaftsdatenbank sie entgegennehmen kann, ohne dass die Transaktionsverarbeitung in der Primärdatenbank blockiert wird.

  Anmerkung: Durch einen größeren Protokollempfangspuffer können Transaktionslastspitzen auf der Primärdatenbank besser ausgeglichen werden, der Puffer wird jedoch trotzdem voll, wenn die durchschnittliche Wiederholungsrate auf der Bereitschaftsdatenbank niedriger ist als die Protokollierungsrate auf der Primärdatenbank.

DB2_HADR_NO_IP_CHECK

- Betriebssystem: Alle
- Standardwert: OFF, Werte: ON | OFF
- Gibt an, ob die IP-Überprüfung für HADR-Verbindungen umgangen werden soll.
- Diese Variable wird vorwiegend in NAT-Umgebungen (NAT = Network Address Translation) verwendet, um die IP-Überprüfung von HADR-Verbindungen zu umgehen. Die Verwendung dieser Variablen in anderen Umgebungen wird nicht empfohlen, weil dadurch die Prüfung auf ordnungsgemäßen Zustand der HADR-Konfiguration beeinträchtigt wird. Standardmäßig wird die Konsistenz der Konfiguration der lokalen und fernen Hostparameter beim Herstellen einer HADR-Verbindung

überprüft. Zur Überprüfung wird eine Zuordnung zwischen Hostnamen und IP-Adressen hergestellt. Dabei werden zwei Überprüfungsoperationen ausgeführt:

- Parameter **HADR_LOCAL_HOST** auf Primäreinheit = Parameter **HADR_REMOTE_HOST** auf Bereitschaftseinheit
- Parameter **HADR_REMOTE_HOST** auf Primäreinheit = Parameter **HADR_LOCAL_HOST** auf Bereitschaftseinheit

Die Verbindung wird geschlossen, wenn bei dieser Überprüfung Fehler festgestellt werden.

Wenn dieser Parameter aktiviert wird, dann führt das System keine IP-Überprüfung durch.

DB2_HADR_PEER_WAIT_LIMIT

- Betriebssystem: Alle
- Standardwert: **0** (d. h. kein Grenzwert), Werte: 0 bis maximal 32-Bit-Integer (einschließlich) ohne Vorzeichen
- Durch das Setzen der Registrierdatenbankvariablen **DB2_HADR_PEER_WAIT_LIMIT** wird für die HADR-Primärdatenbank der Peerstatus aufgehoben, wenn die Protokollierung in der Primärdatenbank für die angegebene Anzahl von Sekunden blockiert wurde, weil das Protokoll in die Bereitschaftsdatenbank repliziert wird. Wenn dieser Grenzwert erreicht wird, unterbricht die Primärdatenbank die Verbindung zur Bereitschaftsdatenbank. Wenn das Peerfenster inaktiviert ist, befindet sich die Primärdatenbank im Status 'unterbrochen' und die Protokollierung wird wieder aufgenommen. Wenn das Peerfenster aktiviert ist, befindet sich die Primärdatenbank im Status 'Unterbrochener Peer', in dem die Protokollierung weiterhin blockiert ist. Die Primärdatenbank verlässt den Status 'Unterbrochener Peer', wenn die Verbindung wieder hergestellt wird oder das Peerfenster abläuft. Die Protokollierung wird wieder aufgenommen, sobald die Primärdatenbank den Status 'Unterbrochener Peer' verlässt. Dieser Parameter ist in der Bereitschaftsdatenbank nicht wirksam. Es empfiehlt sich jedoch, dass in der Primär- und der Bereitschaftsdatenbank der gleiche Wert verwendet wird. Ungültige Werte (keine Zahl oder negative Zahlen) werden als "0" interpretiert, d. h. es gibt keinen Grenzwert. Dieser Parameter ist statisch. Die Datenbankinstanz muss erneut gestartet werden, um diesen Parameter zu aktivieren.

DB2_HADR_ROS

- Betriebssystem: Alle
- Standardwert: OFF Werte: OFF oder ON
- Durch diese Variable wird die Funktion zur Ausführung von HADR-Leseoperationen in der Bereitsschaftsdatenbank aktiviert. Wenn die Variable **DB2_HADR_ROS** auf der HADR-Bereitschaftsdatenbank aktiviert ist, akzeptiert die Bereitschaftsdatenbank Clientverbindungen und lässt die Ausführung reiner Leseabfragen zu. **DB2_HADR_ROS** ist eine statische Registrierdatenbankvariable, sodass nach Änderung der Einstellung die DB2-Instanz erneut gestartet werden muss, um die Änderung in Kraft zu setzen.

DB2_HADR_SORCVBUF

- Betriebssystem: Alle
- Standardwert: TCP-Socketempfangspuffergröße des Betriebssystems, Werte: 1024 bis 4294967295

- Diese Variable gibt für die HADR-Verbindung die Größe des Empfangspuffers für den TCP-Socket des Betriebssystems an und ermöglicht es Benutzern damit, die HADR-TCP/IP-Funktionsweise separat von anderen Verbindungen anzupassen. Bei manchen Betriebssystemen wird der vom Benutzer angegebene Wert automatisch gerundet oder begrenzt. Die tatsächliche Puffergröße für die HADR-Verbindung wird in den Protokolldateien **db2diag** protokolliert. Im Handbuch zur Netzoptimierung für das verwendete Betriebssystem finden Sie Informationen zu den optimalen Einstellungen für diesen Parameter auf der Basis des anfallenden Netzdatenverkehrs. Diese Variable sollte zusammen mit **DB2_HADR_SOSNDBUF** verwendet werden.

DB2_HADR_SOSNDBUF

- Betriebssystem: Alle
- Standardwert: TCP-Socketsendepuffergröße des Betriebssystems, Werte: 1024 bis 4294967295
- Diese Variable gibt für die HADR-Verbindung die Größe des Sendepuffers für den TCP-Socket des Betriebssystems an und ermöglicht es Benutzern damit, die HADR-TCP/IP-Funktionsweise separat von anderen Verbindungen anzupassen. Bei manchen Betriebssystemen wird der vom Benutzer angegebene Wert automatisch gerundet oder begrenzt. Die tatsächliche Puffergröße für die HADR-Verbindung wird in den Protokolldateien **db2diag** protokolliert. Im Handbuch zur Netzoptimierung für das verwendete Betriebssystem finden Sie Informationen zu den optimalen Einstellungen für diesen Parameter auf der Basis des anfallenden Netzdatenverkehrs. Diese Variable sollte zusammen mit **DB2_HADR_SORCVBUF** verwendet werden.

DB2_INDEX_PCTFREE_DEFAULT

- Betriebssystem: Alle
- Standardwert: nicht definiert, Werte: 0 bis 99
- Diese Registerdatenbankvariable gibt an, welcher Prozentsatz der einzelnen Indexseiten bei der Indexerstellung als freier Speicherbereich beibehalten werden soll. Die Einstellung für **DB2_INDEX_PCTFREE_DEFAULT** wird überschrieben, wenn Sie die Klausel PCTFREE explizit in der Anweisung CREATE INDEX angeben. Die Registrierdatenbankvariable hat keine Auswirkung auf die Klausel LEVEL2 PCTFREE der Anweisung CREATE INDEX.

  Die Registrierdatenbankvariable wird nicht zum Zeitpunkt eines Datenbankupgrades angewendet, auch wenn die Indizes während des Upgrades erneut erstellt werden. Sie wird nur auf eine neue Installation angewendet oder nachdem die Durchführung des Upgrades abgeschlossen wurde. Diese Registrierdatenbankvariable ist dynamisch. Sie können sie definieren bzw. ihren Wert löschen, ohne die Instanz stoppen und starten zu müssen.

  Wenn für **DB2_WORKLOAD** der Wert SAP definiert wird, wird **DB2_INDEX_PCTFREE_ADMIN_DEFAULT** auf den Wert 0 gesetzt.

DB2LDAP_BASEDN

- Betriebssystem: Alle
- Standardwert: NULL, Werte: alle gültigen definierten Basisdomänennamen.
- Wenn diese Variable definiert ist, werden die LDAP-Objekte für DB2 im LDAP-Verzeichnis unter

```
CN=System
CN=IBM
CN=DB2
```

unter dem angegebenen definierten Basisnamen (Basis-DN) gespeichert.

Wenn diese Variable für Microsoft Active Directory Server verwendet wird, stellen Sie sicher, dass die Werte CN=DB2, CN=IBM und CN=System unter diesem definierten Namen (DN) definiert sind.

- Änderungen an dieser Variable werden sofort wirksam und gelten für alle zukünftigen kompilierten SQL-Anweisungen. Es ist nicht nötig, die Instanz erneut zu starten oder den Befehl **db2set** mit dem Parameter **-immediate** abzusetzen.

DB2LDAPCACHE

- Betriebssystem: Alle
- Standardwert: `YES`, Werte: `YES` oder `NO`.
- Gibt an, dass der LDAP-Cache aktiviert werden soll. Dieser Cache wird zum Katalogisieren der Datenbank-, Knoten- und DCS-Verzeichnisse auf der lokalen Maschine verwendet.

  Führen Sie die folgenden Befehle aus, um sicherzustellen, dass sich in Ihrem Cache die aktuellsten Einträge befinden:

  ```
  REFRESH LDAP IMMEDIATE ALL
  ```

  Dieser Befehl aktualisiert das Datenbank- und das Knotenverzeichnis und entfernt falsche Einträge.

DB2LDAP_CLIENT_PROVIDER

- Betriebssystem: Windows
- Standardwert: `NULL` (wenn verfügbar, wird `Microsoft` verwendet, andernfalls wird `IBM` verwendet). Werte: `IBM` oder `Microsoft`.
- Bei der Ausführung in einer Windows-Umgebung unterstützt DB2 die Verwendung von Microsoft-LDAP-Clients oder IBM LDAP-Clients zum Zugriff auf das LDAP-Verzeichnis. Diese Registrierdatenbankvariable wird dazu verwendet, den von DB2 zu verwendenden LDAP-Client explizit auszuwählen.

  Anmerkung: Verwenden Sie zum Anzeigen des aktuellen Werts dieser Registrierdatenbankvariablen den Befehl **db2set**:

  ```
  db2set DB2LDAP_CLIENT_PROVIDER
  ```

DB2LDAPHOST

- Betriebssystem: Alle
- Standardwert: Null, Werte: *`basisdomänenname[:portnummer]`* oder *`basisdomänenname`*`:SSL:636` bei Verwendung eines für SSL aktivierten LDAP-Hosts
- Gibt den Hostnamen und die optionale Portnummer der Position des LDAP-Verzeichnisses an. Dabei ist *basisdomänenname* der TCP/IP-Hostname und *[:portnummer]* die Portnummer.
- Änderungen an dieser Variable werden sofort wirksam und gelten für alle zukünftigen kompilierten SQL-Anweisungen. Es ist nicht nötig, die Instanz erneut zu starten oder den Befehl **db2set** mit dem Parameter **-immediate** abzusetzen.

DB2LDAP_KEEP_CONNECTION

- Betriebssystem: Alle

- Standardwert: YES, Werte: YES oder NO.
- Gibt an, ob DB2 die internen LDAP-Verbindungskennungen in den Cache stellt. Wenn diese Variable auf NO gesetzt wird, stellt DB2 die Kennungen für LDAP-Verbindungen zum Verzeichnisserver nicht in den Cache. Dies führt mit einiger Wahrscheinlichkeit zu einer Leistungsbeeinträchtigung. Es kann jedoch wünschenswert sein, die Variable **DB2LDAP_KEEP_CONNECTION** auf den Wert NO zu setzen, wenn die Anzahl simultan aktiver LDAP-Clientverbindungen zum Verzeichnisserver auf ein Minimum reduziert werden muss.

  Diese Variable wird standardmäßig auf den Wert YES gesetzt, um die optimale Leistung zu erzielen.

  Die Registrierdatenbankvariable **DB2LDAP_KEEP_CONNECTION** ist nur als Profilregistrierdatenbankvariable der globalen Ebene in LDAP implementiert, sodass sie mit dem Befehl **db2set** unter Angabe der Option **-gl** wie folgt definiert werden muss:

  ```
  db2set -gl DB2LDAP_KEEP_CONNECTION=NO
  ```

DB2LDAP_SEARCH_SCOPE

- Betriebssystem: Alle
- Standardwert: DOMAIN, Werte: LOCAL, DOMAIN oder GLOBAL
- Gibt den Bereich für die Suche nach Informationen an, die sich in Datenbankpartitionen oder Domänen im Lightweight Directory Access Protocol (LDAP) befinden. Der Wert LOCAL inaktiviert das Suchen im LDAP-Verzeichnis. Beim Wert DOMAIN wird das LDAP-Verzeichnis nur nach der aktuellen Verzeichnispartition durchsucht. Beim Wert GLOBAL wird das LDAP-Verzeichnis in allen Verzeichnispartitionen durchsucht, bis das Objekt gefunden wird.

DB2_LIMIT_FENCED_GROUP

- Betriebssystem: Windows
- Standardwert: NULL, Werte: ON oder OFF
- Wenn die erweiterte Sicherheit aktiviert ist, können Sie die Zugriffsrechte des Betriebssystems für den Prozess im abgeschirmten Modus (**db2fmp**) auf die Zugriffsrechte einschränken, die der Gruppe DB2USERS zugeordnet sind. Dazu setzen Sie diese Registrierdatenbankvariable auf den Wert ON und fügen das DB2-Servicekonto (d. h. den Benutzernamen, unter dem der DB2-Service ausgeführt wird) der Gruppe DB2USERS hinzu.

  Anmerkung: Wenn das lokale Systemkonto (LocalSystem) als DB2-Servicekonto verwendet wird, hat die Einstellung der Variablen **DB2_LIMIT_FENCED_GROUP** keine Wirkung.

  Sie können dem Prozess **db2fmp** zusätzliche Betriebssystemzugriffsrechte erteilen, indem Sie das DB2-Servicekonto einer Betriebssystemgruppe hinzufügen, die diese zusätzlichen Zugriffsrechte besitzt.

DB2_LOAD_COPY_NO_OVERRIDE

- Betriebssystem: Alle
- Standardwert: NONRECOVERABLE, Werte: COPY YES oder NONRECOVERABLE
- Diese Variable konvertiert jeden Befehl **LOAD COPY NO** je nach Wert der Variablen entweder in **LOAD COPY YES** oder **NONRECOVERABLE**. Diese Variable gilt für HADR-Primärdatenbanken und für Standarddatenbanken (nicht HADR). Sie wird in einer HADR-Bereitschaftsdatenbank ignoriert. Wenn in einer HADR-Primärdatenbank diese Variable nicht definiert ist, wird der Befehl **LOAD COPY NO** in **LOAD NONRECOVERABLE** konvertiert. Der

Wert dieser Variablen gibt entweder eine nicht wiederherstellbare Ladeoperation oder das Kopierziel an, wobei die gleiche Syntax wie bei der Klausel **COPY YES** zu verwenden ist.

- Änderungen an dieser Variable werden sofort wirksam und gelten für alle zukünftigen kompilierten SQL-Anweisungen. Es ist nicht nötig, die Instanz erneut zu starten oder den Befehl **db2set** mit dem Parameter **-immediate** abzusetzen.

DB2LOADREC

- Betriebssystem: Alle
- Standardwert: NULL
- Dient zum Überschreiben der Speicherposition der Ladekopie bei einer aktualisierenden Recovery (ROLLFORWARD). Wenn der Benutzer die physische Speicherposition der Ladekopie geändert hat, muss die Variable **DB2LOADREC** vor dem Ausführen der aktualisierenden Recovery definiert werden.
- Änderungen an dieser Variable werden sofort wirksam und gelten für alle zukünftigen kompilierten SQL-Anweisungen. Es ist nicht nötig, die Instanz erneut zu starten oder den Befehl **db2set** mit dem Parameter **-immediate** abzusetzen.

DB2LOCK_TO_RB

- Betriebssystem: Alle
- Standardwert: NULL, Werte: STATEMENT
- Gibt an, ob ein Rollback bei Überschreitungen von Sperrzeiten jeweils für die gesamte Transaktion oder nur für die aktuelle Anweisung durchgeführt werden soll. Wenn **DB2LOCK_TO_RB** den Wert STATEMENT hat, führen Überschreitungen von Sperrzeiten dazu, dass nur die aktuelle Anweisung durch ein Rollback rückgängig gemacht wird. Jeder andere Wert für diesen Parameter bewirkt, dass die gesamte Transaktion durch ein Rollback rückgängig gemacht wird.

DB2_MAX_LOB_BLOCK_SIZE

- Betriebssystem: Alle
- Standardwert: 0 (kein Zeitlimit), Werte: 0 bis 21487483647
- Legt die maximale Größe von LOB- oder XML-Daten fest, die in einem Block zurückgegeben werden sollen. Dieser Wert ist kein fixer Maximalwert. Wenn dieser Maximalwert auf dem Server während eines Datenabrufs erreicht wird, beendet der Server das Schreiben der aktuellen Zeile, bevor er eine Antwort für den Befehl, zum Beispiel FETCH, an den Client generiert.

DB2_MEMORY_PROTECT

- Betriebssystem: AIX mit Speicherschlüsselunterstützung
- Standardwert: NO, Werte: NO oder YES
- Diese Registrierdatenbankvariable aktiviert eine Speicherschutzfunktion, die mit Speicherschlüsseln arbeitet, um durch unzulässige Speicherzugriffe verursachte Datenverluste im Pufferpool zu verhindern. Der Speicherschutz funktioniert in der Weise, dass bestimmt wird, zu welchen Zeiten die Threads der DB2-Steuerkomponente Zugriff auf den Pufferpoolspeicher haben sollen und zu welchen sie keinen Zugriff haben sollen. Wenn **DB2_MEMORY_PROTECT** auf YES gesetzt ist, wird jeder unzulässige Versuch eines Threads der DB2-Steuerkomponente, auf den Pufferpoolspeicher zuzugreifen, abgefangen (Trap).

Anmerkung: Sie können keinen Speicherschutz verwenden, wenn **DB2_LGPAGE_BP** auf YES gesetzt ist. Selbst wenn **DB2_MEMORY_PROTECT** auf YES gesetzt ist, wird DB2 den Pufferpoolspeicher nicht schützen und die Funktion inaktivieren.

DB2_MIN_IDLE_RESOURCES

- Betriebssystem: Linux
- Standardwert: OFF, Werte: OFF oder ON
- Diese Variable gibt an, dass eine aktivierte Datenbank ein Minimum an Verarbeitungsressourcen verwenden soll, wenn sie sich im Leerlauf befindet. Dies kann in einigen virtuellen Linux-Umgebungen nützlich sein (z. B. z/VM), in denen die kleinen Ressourceneinsparungen den VM-Hostmonitor dabei unterstützen, die CPU- und Speicherressourcen über alle virtuellen Maschinen hinweg effizienter zu planen.

DB2_NCHAR_SUPPORT

- Betriebssystem: Alle
- Standardwert: ON, Werte: ON oder OFF
- Wenn diese Variable auf ON gesetzt ist (Standardwert), stehen die NCHAR-, NVARCHAR- und NCLOB-Schreibweisen für die Grafikdatentypen zur Verwendung in Unicode-Datenbanken zur Verfügung. Außerdem sind verschiedene Funktionen verfügbar, die sich auf nationale Sonderzeichen beziehen, beispielsweise NCHAR() und TO_NCHAR().

  Diese Variable sollte nur dann auf den Wert OFF gesetzt werden, wenn in einer vorhandenen Datenbank benutzerdefinierte Typen namens NCHAR, NVARCHAR oder NCLOB verwendet werden.

  Anmerkung: Die Registrierdatenbankvariable **DB2_NCHAR_SUPPORT** wird in einem künftigen Release eventuell entfernt. Ab diesem Zeitpunkt ist es nicht mehr möglich, in der Datenbank benutzerdefinierten Typen namens NCHAR, NVCHAR oder NCLOB zu verwenden.

DB2NOEXITLIST

- Betriebssystem: Alle
- Standardwert: OFF, Werte: ON oder OFF
- Diese Variable gibt an, dass DB2 keine Exitlistenroutine laden darf und dass unabhängig von der Einstellung der Registrierdatenbankvariablen **DB2_COMMIT_ON_EXIT** keine Commitoperation ausgeführt werden darf, wenn die Anwendung beendet wird.

  Wenn **DB2NOEXITLIST** inaktiviert und **DB2_COMMIT_ON_EXIT** aktiviert ist, werden alle unvollständigen Transaktionen für Anwendungen mit eingebettetem SQL automatisch festgeschrieben. Sie sollten explizit COMMIT- oder ROLLBACK-Anweisungen hinzufügen, wenn eine Anwendung beendet wird.

  Anwendungen, die vor der Beendigung der Anwendung die DB2-Bibliothek dynamisch laden und entladen, stürzen möglicherweise ab, wenn die DB2-Exitroutine aufgerufen wird. Dieser Absturz kann darauf zurückzuführen sein, dass die Anwendung versucht, eine Funktion aufzurufen, die im Speicher nicht vorhanden ist. Um diese Situation zu vermeiden, muss die Registrierdatenbankvariable **DB2NOEXITLIST** gesetzt werden.

DB2_NUM_CKPW_DAEMONS

- Betriebssystem: UNIX

- Standardwert: 3, Werte: 1[:FORK] bis 100[:FORK]
- Mithilfe der Registrierdatenbankvariablen **DB2_NUM_CKPW_DAEMONS** können Sie eine konfigurierbare Anzahl von Kennwortprüfdämonen starten. Die Dämonen werden bei Ausführung von **db2start** erstellt und verarbeiten Anforderungen zum Prüfen von Kennwörtern, wenn das Standardsicherheits-Plug-in **IBMOSauthserver** verwendet wird. Eine Erhöhung des Werts von **DB2_NUM_CKPW_DAEMONS** kann die Zeit verkürzen, die zur Herstellung einer Datenbankverbindung benötigt wird. Dies gilt jedoch nur in Szenarios, in denen viele Verbindungen gleichzeitig hergestellt werden und in denen die Authentifizierung aufwendig ist.

  Die Variable **DB2_NUM_CKPW_DAEMONS** kann auf einen Wert zwischen 1 und 100 gesetzt werden. Der Datenbankmanager erstellt die Anzahl von Dämonen, die durch **DB2_NUM_CKPW_DAEMONS** angegeben wird. Jeder Dämon kann Anforderungen zum Prüfen von Kennwörtern direkt verarbeiten.

  Der optionale Parameter FORK kann hinzugefügt werden, um die Dämonen zum Prüfen von Kennwörtern zu befähigen, ein externes Kennwortprüfprogramm (**db2ckpw**) zur Verarbeitung von Anforderungen zum Prüfen von Kennwörtern explizit zu starten. Dies ist der Einstellung von **DB2_NUM_CKPW_DAEMONS** auf den Wert null in früheren Releases ähnlich. Im Modus FORK startet jeder Dämon zum Prüfen von Kennwörtern das Kennwortprüfprogramm für jede Anforderung zum Prüfen eines Kennworts. Die Dämonen im Modus FORK werden als Instanzeigner gestartet.

  Wenn **DB2_NUM_CKPW_DAEMONS** auf den Wert null gesetzt wird, wird der effektive Wert auf 3:FORK gesetzt, sodass drei Dämonen zum Prüfen von Kennwörtern im Modus FORK gestartet werden.

DB2_OPTSTATS_LOG

- Betriebssystem: Alle
- Standardwert: nicht definiert (Details siehe unten), Werte: OFF, ON {NUM | SIZE | NAME | DIR}
- Die Variable **DB2_OPTSTATS_LOG** gibt die Attribute der Protokolldateien für Statistikereignisse an, die zum Überwachen und Analysieren der Statistikerfassung in Bezug auf Aktivitäten verwendet werden. Wenn **DB2_OPTSTATS_LOG** nicht definiert oder auf ON gesetzt wird, wird die Protokollierung von Statistikereignissen aktiviert, sodass Sie zur besseren Fehlerbestimmung die Systemleistung überwachen und ein Verlaufsprotokoll anlegen können. Die Protokollsätze werden in die erste Protokolldatei geschrieben, bis diese gefüllt ist. Nachfolgende Protokollsätze werden in die nächste verfügbare Protokolldatei geschrieben. Wenn die maximale Anzahl von Dateien erreicht ist, wird die älteste Protokolldatei mit den neuen Protokollsätzen überschrieben. Wenn die Systemressourcennutzung in Ihrer Umgebung problematisch ist, können Sie diese Registrierdatenbankvariable inaktivieren, indem Sie sie auf OFF setzen.
- Änderungen an dieser Variable werden sofort wirksam und gelten für alle zukünftigen kompilierten SQL-Anweisungen. Es ist nicht nötig, die Instanz erneut zu starten oder den Befehl **db2set** mit dem Parameter **-immediate** abzusetzen.

  Wenn die Protokollierung von Statistikereignissen explizit aktiviert (auf ON gesetzt) wird, können Sie eine Reihe von Optionen ändern:

- NUM: Die maximale Anzahl rollierender Protokolldateien. Standardwert: 5, Werte: 1 - 15

- SIZE: Die maximale Größe der rollierenden Protokolldateien. (Die Größe jeder einzelnen rollierenden Datei ist SIZE/NUM.) Standardwert= 15 MM, Werte: 1 MB – 4096 MB
- NAME: Der Basisname für rollierende Protokolldateien. Standardwert: db2optstats.*nummer*.log, z. B. db2optstats.0.log, db2optstats.1.log usw.
- DIR: Das Basisverzeichnis für rollierende Protokolldateien. Standardwert: **diagpath**/events

Sie können eine beliebige Anzahl dieser Optionen angegeben. Sie müssen nur sicherstellen, dass Sie als ersten Wert ON angeben, wenn Sie die Statistikprotokollierung aktivieren wollen. Wenn Sie zum Beispiel die Statistikprotokollierung mit einer maximalen Anzahl von 6 Protokolldateien, einer maximalen Dateigröße von 25 MB sowie dem Basisdateinamen mystatslog und dem Verzeichnis mystats aktivieren wollen, geben Sie den folgenden Befehl ein:

```
db2set DB2_OPTSTATS_LOG=ON,NUM=6,SIZE=25,NAME=mystatslog,DIR=mystats
```

DB2REMOTEPREG

- Betriebssystem: Windows
- Standardwert: NULL, Werte: beliebiger gültiger Windows-Computername
- Definiert den Namen des fernen Computers, der die Win32-Registrierdatenbankliste von DB2-Instanzprofilen und DB2-Instanzen enthält. Der Wert für **DB2REMOTEPREG** darf erst nach der Installation des DB2-Datenbankprodukts definiert werden und darf nach der Definition nicht mehr geändert werden. Verwenden Sie diese Variable mit großer Vorsicht.
- In einer Umgebung mit partitionierten Datenbanken können Sie die Registrierdatenbankvariable **DB2REMOTEPREG** verwenden, um einen Computer, der nicht der Instanzeigner ist, so zu konfigurieren, dass er die Werte von Registrierdatenbankvariablen des Computers verwendet, der Instanzeigner ist. Weitere Informationen zu den Situationen, in denen diese Variable verwendet werden sollte, finden Sie im Abschnitt „Definieren von Variablen auf Instanzebene in einer Umgebung mit partitionierten Datenbanken” auf Seite 625.

  Wenn der DB2-Datenbankmanager die Registrierdatenbankvariablen unter Windows-Betriebssystemen liest, liest er zuerst den Wert von **DB2REMOTEPREG**. Wenn die Variable **DB2REMOTEPREG** definiert ist, öffnet der Datenbankmanager die Registrierdatenbank auf dem fernen Computer, der in der Variablen **DB2REMOTEPREG** angegeben ist. Nachfolgende Lese- und Aktualisierungsoperationen für die Registrierdatenbankvariablen werden an den angegebenen fernen Computer weitergeleitet.

  Wenn ein Computer, der nicht der Instanzeigner ist, auf die ferne Registrierdatenbank zugreifen soll, muss auf dem Zielcomputer der Service für die Fernregistrierung (Remote Registry Service) aktiv sein. Außerdem müssen das Benutzeranmeldekonto und alle DB2-Serviceanmeldekonten über ausreichende Zugriffsrechte für die ferne Registrierdatenbank verfügen. Bei der Verwendung der Variablen **DB2REMOTEPREG** müssen Sie daher in einer Windows-Domänenumgebung arbeiten, sodass Sie dem Domänenkonto den erforderlichen Registrierdatenbankzugriff erteilen können.
- Verwenden Sie die Variable **DB2REMOTEPREG** nicht in einer Microsoft Cluster Server-Umgebung.

DB2_RESOLVE_CALL_CONFLICT

- Betriebssystem: AIX, HP-UX, Solaris, Linux, Windows

- Standardwert: YES, Werte: YES, NO
- Wenn durch Trigger aufgerufene Routinen versuchen, auf Tabellen zuzugreifen, die durch andere Anweisungen oder Routinen im Hauptteil des Triggers geändert wurden, kann dies verschachtelte SQL-Anweisungsregeln verletzen. Wenn **DB2_RESOLVE_CALL_CONFLICT** eingestellt wird, erzwingt dies die Ausführung sämtlicher Änderungen an Tabellen entsprechend den SQL-Standardregeln für Trigger, bevor die Anweisung CALL ausgeführt wird..

  Sie müssen die Instanz stoppen, bevor Sie **DB2_RESOLVE_CALL_CONFLICT** neu definieren, und sie anschließend neu starten. Binden Sie anschließend sämtliche Pakete neu, die den Aufruf von Triggern verursachen. SQL-Prozeduren binden Sie wie folgt erneut: CALL SYSPROC.REBIND_ROUTINE_PACKAGE ('P','*prozedurschema.prozedurname*','CONSERVATIVE');

  Beachten Sie, dass sich **DB2_RESOLVE_CALL_CONFLICT** auf die Leistung auswirken kann. Wenn **DB2_RESOLVE_CALL_CONFLICT** auf YES gesetzt wird, löst der DB2-Datenbankmanager alle potenziellen Lese- und Schreibkonflikte durch Einschaltung temporärer Tabellen (nach Bedarf). In der Regel ist die Wirkung gering, da maximal eine temporäre Tabelle eingefügt wird. Die Wirkung in einer OLTP-Umgebung ist gering, da nur eine Zeile bzw. eine kleine Anzahl von Zeilen durch die Triggeranweisung geändert wird. In der Regel ist der Einfluss auf die Leistung durch **DB2_RESOLVE_CALL_CONFLICT** gering, wenn die allgemeinen Empfehlungen zur Verwendung von SMS (systemverwalteter Speicherbereich, System Managed Storage) für temporäre Tabellenbereiche befolgt werden.
- Änderungen an dieser Variablen werden sofort wirksam und gelten für alle zukünftig kompilierten SQL-Anweisungen, wenn der Befehl **db2set** mit dem Parameter **-immediate** ausgegeben wird. Es ist nicht erforderlich, die Instanz erneut zu starten.

DB2SATELLITEID

- Betriebssystem: Alle
- Standardwert: NULL, Werte: eine gültige Satelliten-ID, die in der Satellitensteuerungsdatenbank deklariert ist
- Gibt die Satelliten-ID an, die an den Satellitensteuerungsserver übergeben wird, wenn ein Satellit eine Synchronisation durchführt. Wenn für diese Variable kein Wert angegeben ist, wird die Anmelde-ID als Satelliten-ID verwendet.

DB2_SERVER_CONTIMEOUT

- Betriebssystem: Alle
- Standardwert: 180, Werte: 0 bis 32767 Sekunden
- Diese Registrierdatenbankvariable und die Registrierdatenbankvariable **DB2_DISPATCHER_PEEKTIMEOUT** dienen beide zur Konfiguration der Behandlung eines neuen Clients während der Verbindungszeit. Mit **DB2_SERVER_CONTIMEOUT** können Sie die Zeit (in Sekunden) anpassen, die ein Agent auf die Verbindungsanforderung eines Clients wartet, bevor er die Verbindung beendet. In den meisten Fällen sollte es nicht nötig sein, diese Registrierdatenbankvariable anzupassen. Wenn DB2-Clients beim Verbindungsaufbau jedoch beständig das vom Server definierte Zeitlimit überschreiten, können Sie einen höheren Wert für **DB2_SERVER_CONTIMEOUT** angeben, um das Zeitlimitintervall zu verlängern. Wenn ein ungültiger Wert definiert wird, wird der Standardwert verwendet. Diese Registrierdatenbankvariable ist nicht dynamisch.

DB2_SERVER_ENCALG

- Betriebssystem: Alle
- Standardwert: NULL, Werte: AES_CMP oder AES_ONLY
-

  Anmerkung: DB2_SERVER_ENCALG ist in Version 9.7 veraltet und wird in einem zukünftigen Release möglicherweise entfernt.

  Ist die Registrierdatenbankvariable **DB2_SERVER_ENCALG** bei einem Upgrade der Instanzen auf DB2 Version 9.7 definiert, wird der Konfigurationsparameter **alternate_auth_enc** je nach Einstellung für **DB2_SERVER_ENCALG** mit AES_ONLY oder AES_CMP definiert. Aktualisieren Sie anschließend den Konfigurationsparameter **alternate_auth_enc**, um den Verschlüsselungsalgorithmus für die Verschlüsselung von Benutzer-IDs und Kennwörtern anzugeben. Wird der Konfigurationsparameter **alternate_auth_enc** definiert, überschreibt der zugehörige Wert den Wert für die Registrierdatenbankvariable **DB2_SERVER_ENCALG**.

DB2SORT

- Betriebssystem: Alle, nur Server
- Standardwert: NULL
- Diese Variable gibt die Speicherposition einer während der Laufzeit durch das Dienstprogramm LOAD zu ladenden Bibliothek an. Die Bibliothek enthält den Eingangspunkt für Funktionen, die beim Sortieren von Indexdaten verwendet werden. Verwenden Sie **DB2SORT**, um Sortierprogrammprodukte anderer Lieferanten mit dem Dienstprogramm LOAD zur Generierung von Tabellenindizes zu nutzen. Der angegebene Pfad muss relativ zum Datenbankserver definiert werden.

DB2_STANDBY_ISO

- Betriebssystem: Alle
- Standardwert: NULL, Werte: UR oder OFF
- Diese Variable legt die Isolationsstufe, die von Anwendungen und Anweisungen angefordert wird, die in einer aktiven HADR-Bereitschaftsdatenbank ausgeführt werden, zwingend auf UR (Uncommitted Read, nicht festgeschriebener Lesevorgang) fest. Wenn die Variable **DB2_STANDBY_ISO** auf den Wert UR gesetzt wird, werden höhere Isolationsstufen als UR auf UR herabgestuft, ohne dass eine Warnung zurückgegeben wird. Wenn die HADR-Bereitschaftsdatenbank die Funktion als HADR-Primärdatenbank übernimmt, hat diese Variable keine Wirkung.

DB2STMM

- Betriebssystem: UNIX
- Diese Registrierdatenbankvariable steuert eine Gruppe von Parametern, mit denen Sie bestimmte Merkmale des Speichermanagers für automatische Leistungsoptimierung (STMM, Self-Tuning Memory Manager) ändern können.
- Parameter:

  GLOBAL_BENEFIT_SEGMENT_COMPATIBLE

  – Standardwert: nicht definiert, Werte: YES, NO

  – Der Parameter GLOBAL_BENEFIT_SEGMENT_COMPATIBLE hat nur dann Auswirkungen auf die Funktionsweise, wenn der Konfigurationsparameter **database_memory** für eine Datenbank auf AUTOMATIC gesetzt ist.

Dieser Parameter beeinflusst die Berechtigungseinstellungen des gemeinsam genutzten Speichersegments des STMM. Er darf nur auf Systemen mit mehreren Instanzen auf YES gesetzt werden, auf dem einige der Instanzen eine ältere Version und für **database_memory** den Wert AUTOMATIC aufweisen, um Probleme bei der Abwärtskompatibilität zu mindern, die sich auf die Optimierung der allgemeinen Speichernutzung einer Datenbank auswirken. Eine ältere Instanz würde beispielsweise zu einem der folgenden DB2-Releases und Fixpackstufen gehören: DB2 V9.1 mit allen Fixpackstufen, DB2 V9.5 Fixpack 7 und früher sowie DB2 V9.7 Fixpack 4 und früher.

Bei Instanzen, die Nichtrootinstallationen von DB2 sind, sollten Sie diese Variable nur einstellen, wenn alle Instanzen auf dem System dasselbe gemeinsam genutzte STMM-Speichersegment verwenden sollen. Wird diese Variable nicht festgelegt oder mit NO definiert, verwendet eine Nicht-Root-Instanz das eigene, instanzspezifische gemeinsam genutzte STMM-Speichersegment. Dies kann sich auf die Optimierung der allgemeinen Datenbankspeicherauslastung für alle Datenbanken auswirken, für die **database_memory** auf AUTOMATIC gesetzt ist.

Diese Registrierdatenbankvariable wird einmal während des Starts der DB2-Instanz gelesen. Dieser Parameter muss für alle Instanzen, für die ein Upgrade durchgeführt wurde (d. h. die nicht auf einem Vorversionsstand sind), festgelegt werden; nach dem Festlegen des Parameters müssen alle Instanzen, für die ein Upgrade durchgeführt wurde, neu gestartet werden.

GLOBAL_BENEFIT_SEGMENT_UNIQUE

- Standardwert: nicht definiert, Werte: YES, NO
- Der Parameter GLOBAL_BENEFIT_SEGMENT_UNIQUE hat nur dann Auswirkungen auf die Funktionsweise, wenn der Konfigurationsparameter **database_memory** für eine Datenbank auf AUTOMATIC gesetzt ist.

  Dieser Parameter gibt an, dass jede Instanz, für die ein Upgrade durchgeführt wurde (d. h. die nicht auf einem Vorversionsstand ist), das eigene, instanzspezifische gemeinsam genutzte STMM-Speichersegment verwenden soll. Dies bedeutet, dass jede Instanz die allgemeine Datenbankspeicherauslastung für jede ihr zugeordnete Datenbank optimiert, unabhängig von der Optimierung der allgemeinen Datenbankspeicherauslastung für Datenbanken, die anderen Instanzen im System zugeordnet sind.

  Setzen Sie diesen Parameter auf YES, wenn der Konfigurationsparameter **instance_memory nicht** für alle Instanzen in einem System auf AUTOMATIC gesetzt ist.

  Diese Registrierdatenbankvariable wird einmal während des Starts der DB2-Instanz gelesen. Dieser Parameter muss für alle Instanzen, für die ein Upgrade durchgeführt wurde, festgelegt werden; nach dem Festlegen des Parameters müssen alle Instanzen, für die ein Upgrade durchgeführt wurde, neu gestartet werden.

DB2_TRUNCATE_REUSESTORAGE

- Betriebssystem: Alle
- Standardwert: NULL (nicht definiert), Werte: IMPORT, import

- Mithilfe dieser Variablen können Sie Sperrenkonflikte zwischen dem Befehl **IMPORT** mit **REPLACE** und dem Befehl **BACKUP ... ONLINE** auflösen. In einigen Fällen können ein Online-Backup und Abschneideoperationen (TRUNCATE) nicht gleichzeitig ausgeführt werden. Wenn dieser Fall eintritt, können Sie die Variable **DB2_TRUNCATE_REUSESTORAGE** auf den Wert IMPORT bzw. import setzen. Dadurch wird das physische Abschneiden des Objekts, einschließlich Daten, Indizes, Langfelddaten, große Objekte (LOBs) und Blockzuordnungen (für MDC-Tabellen) übersprungen und nur ein logisches Abschneiden ausgeführt. Das heißt, der Befehl **IMPORT** mit **REPLACE** leert die Tabelle, sodass sich die logische Größe des Objekts verringert, der Plattenspeicherplatz jedoch zugeordnet bleibt.

  Diese Registrierdatenbankvariable ist dynamisch. Sie können sie definieren bzw. ihren Wert löschen, ohne die Instanz stoppen und starten zu müssen. Sie können die Variable **DB2_TRUNCATE_REUSESTORAGE** vor dem Start eines Online-Backups definieren und nach Abschluss des Online-Backups ihren Wert wieder löschen. In Umgebungen mit mehreren Datenbankpartitionen ist die Registrierdatenbankvariable nur auf den Knoten aktiv, auf denen sie definiert ist. Die Variable **DB2_TRUNCATE_REUSESTORAGE** ist nur auf permanenten DMS-Objekten wirksam.

  Wenn in SAP-Umgebungen **DB2_WORKLOAD**=SAP definiert ist, hat diese Registrierdatenbankvariable den Standardwert IMPORT.
- Änderungen an dieser Variable werden sofort wirksam und gelten für alle zukünftigen kompilierten SQL-Anweisungen. Es ist nicht nötig, die Instanz erneut zu starten oder den Befehl **db2set** mit dem Parameter **-immediate** abzusetzen.

DB2_UTIL_MSGPATH

- Betriebssystem: Alle
- Standardwert: Verzeichnis *instanzname*/tmp
- Die Registrierdatenbankvariable **DB2_UTIL_MSGPATH** wird in Verbindung mit der Prozedur SYSPROC.ADMIN_CMD, der Prozedur SYSPROC.ADMIN_REMOVE_MSGS und der benutzerdefinierten Funktion (UDF) SYSPROC.ADMIN_GET_MSGS verwendet. Sie gilt für die Instanzebene. Die Variable **DB2_UTIL_MSGPATH** kann definiert werden, um einen Verzeichnispfad auf dem Server anzugeben, in dem die abgeschirmte Benutzer-ID Dateien lesen, schreiben und löschen kann. Dieses Verzeichnis muss von allen Koordinatorpartitionen aus zugänglich sein und genügend Speicherplatz besitzen, um die Nachrichtendateien von Dienstprogrammen aufzunehmen.

  Wenn dieser Pfad nicht festgelegt ist, wird standardmäßig das Verzeichnis *instanzname*/tmp verwendet. (Beachten Sie, dass das Verzeichnis *instanzname*/tmp bei der Deinstallation von DB2 bereinigt wird.)

  Wenn dieser Pfad bei der Ausführung der Prozedur ALTOBJ nicht definiert ist, wird im Verzeichnis ~sqllib/tmp eine temporäre Nachrichtendatei erstellt.

  Wenn dieser Pfad geändert wird, werden die Dateien, die sich in dem durch die vorige Einstellung der Variablen bezeichneten Verzeichnis befanden, nicht automatisch versetzt oder gelöscht. Wenn Sie den Inhalt der Nachrichten, die unter dem alten Pfad erstellt wurden, abrufen wollen, müssen Sie diese Nachrichten (die den Dienstprogrammnamen als Präfix und die Benutzer-ID als Suffix haben) manuell in das neue Ver-

zeichnis versetzen, das durch **DB2_UTIL_MSGPATH** angegeben wird. Die nächste Nachrichtendatei eines Dienstprogramms wird an der neuen Position erstellt, gelesen und bereinigt.

Die Dateien unter dem durch **DB2_UTIL_MSGPATH** angegebenen Verzeichnis sind für die jeweiligen Dienstprogramme spezifisch und nicht transaktionsabhängig. Sie sind kein Teil des Backup-Images. Die Dateien unter dem durch **DB2_UTIL_MSGPATH** angegebenen Verzeichnis werden vom Benutzer verwaltet. Dies bedeutet, dass ein Benutzer die Nachrichtendateien mithilfe der Prozedur SYSPROC.ADMIN_REMOVE_UTILMSG löschen kann. Diese Dateien werden bei der Deinstallation von DB2 nicht bereinigt.

DB2_XBSA_LIBRARY

- Betriebssystem: AIX, HP-UX, Solaris und Windows
- Standardwert: NULL, Werte: eine beliebige gültige Angabe für Pfad und Datei.
- Diese Registrierdatenbankvariable verweist auf die vom Hersteller bereitgestellte XBSA-Bibliothek. Unter AIX muss die Einstellung das gemeinsam genutzte Objekt enthalten, sofern es nicht den Namen shr.o hat. Für HP-UX, Solaris und Windows wird der Name des gemeinsam genutzten Objekts nicht benötigt. Wenn zum Beispiel NetWorker Business Suite Module für DB2 von Legato verwendet werden soll, muss diese Registrierdatenbankvariable wie folgt eingestellt werden:

  ```
  db2set DB2_XSBA_LIBRARY="/usr/lib/libxdb2.a(bsashr10.o)"
  ```

  Die XBSA-Schnittstelle kann mit dem Befehl **BACKUP DATABASE** oder **RESTORE DATABASE** aufgerufen werden. Beispiel:

  ```
  db2 backup db sample use XBSA
  db2 restore db sample use XBSA
  ```

DB2_XSLT_ALLOWED_PATH

- Betriebssystem: Alle
- Standardwert: NULL oder NONE, Werte: ALL oder eine Liste gültiger URIs, die durch Leerzeichen getrennt sind.
- Diese Registrierdatenbankvariable steuert, wie die DB2-Instanz auf die externen Entitäten verweist, die in einem XSLT-Style-Sheet definiert sind. Standardmäßig darf die Dokumentfunktion XSLT nicht auf eine XML-Datei zugreifen. Sie können diese Variable verwenden, um den Zugriff auf Dateien in bestimmten Verzeichnissen zuzulassen, indem Sie einen absoluten Pfad angeben. Wenn Sie mehrere Pfade angeben, müssen Sie diese durch Leerzeichen voneinander trennen. Wenn der Pfad ein Leerzeichen enthält, muss dieses Leerzeichen im Pfad durch die Escapezeichenfolge %20 ersetzt werden.

  Folgende Werte sind für diese Variable zulässig:

  – NULL oder NONE: Es sind keine URI-Verweise zulässig und die Umwandlung mit einem solchen Style-Sheet schlägt fehl.
  – ALL: Alle Verweise auf URIs sind zulässig.

    Anmerkung: Ein nicht kontrollierter Verweis auf einen externen URI kann ein schwerwiegendes Sicherheitsproblem darstellen.

– URI-Liste: Es sind nur Verweise auf URIs zulässig, die sich in Unterverzeichnissen der URIs in der Liste befinden, wie aus folgendem Beispiel hervorgeht:

```
db2set DB2_XSLT_ALLOWED_PATH ="http://some.website.com/test/dir
/home/Joe/resource.txt"
```

Das folgende Beispiel zeigt die Verwendung der Escapezeichenfolge für die Leerzeichen in einem Pfad, der das Verzeichnis `Dokumente und Einstellungen` enthält:

```
db2set DB2_XSLT_ALLOWED_PATH ="file:///C:/
Dokumente%20und%20Einstellungen/joe/xml_files"
```

Kapitel 23. Konfigurationsparameter

Bei der Erstellung einer DB2-Datenbankinstanz oder -Datenbank wird eine entsprechende Konfigurationsdatei mit Standardparameterwerten erstellt. Diese Parameterwerte können zur Verbesserung der Leistung und anderer Merkmale der Instanz oder der Datenbank geändert werden.

Der vom Datenbankmanager auf der Basis von Standardwerten für die Parameter zugeordnete Plattenspeicherplatz und der Hauptspeicher können für Ihre Anforderungen in einigen Fällen ausreichend sein. In einigen Situationen wird jedoch die maximale Leistung bei Verwendung dieser Standardwerte möglicherweise nicht erreicht.

Konfigurationsdateien enthalten Parameter, die Werte definieren, wie zum Beispiel die den DB2-Datenbankprodukten und einzelnen Datenbanken zugeordneten Ressourcen und die Diagnoseebene. Es gibt zwei Arten von Konfigurationsdateien:

- Die Konfigurationsdatei des Datenbankmanagers für jede DB2-Instanz
- Die Datenbankkonfigurationsdatei für jede einzelne Datenbank

Eine *Konfigurationsdatei des Datenbankmanagers* wird erstellt, wenn eine DB2-Instanz erstellt wird. Die in ihr enthaltenen Parameter betreffen Systemressourcen auf Instanzebene, die von keiner Datenbank abhängig sind, die zu dieser Instanz gehört. Die Werte vieler dieser Parameter können von den Systemstandardwerten zur Leistungsoptimierung oder Kapazitätserhöhung abhängig von der Konfiguration des Systems geändert werden.

Darüber hinaus gibt es auch für jede Clientinstallation eine Konfigurationsdatei des Datenbankmanagers. Diese Datei enthält Informationen über den Client Enabler für eine bestimmte Workstation. Für den Client gilt eine Untergruppe der für einen Server verfügbaren Parameter.

Konfigurationsparameter des Datenbankmanagers werden in einer Datei mit dem Namen `db2systm` gespeichert. Diese Datei wird erstellt, wenn die Instanz des Datenbankmanagers erstellt wird. In Linux- und UNIX-Umgebungen befindet sich diese Datei im Unterverzeichnis `sqllib` für die Instanz des Datenbankmanagers. Unter Windows variiert die Standardposition dieser Datei je nach Edition der Windows-Betriebssystemfamilie. Zur Feststellung des Standardverzeichnisses unter Windows prüfen Sie die Einstellung der Registrierdatenbankvariablen **DB2INSTPROF** mithilfe des Befehls **db2set DB2INSTPROF**. Sie können das Standardinstanzverzeichnis auch ändern, indem Sie den Wert der Registrierdatenbankvariablen **DB2INSTPROF** ändern. Wenn die Variable **DB2INSTPROF** definiert ist, befindet sich die Datei im Unterverzeichnis `instance` des Verzeichnisses, das durch die Variable **DB2INSTPROF** angegeben wird.

Andere Profilregistrierdatenbankvariablen, die angeben, wo sich die Laufzeitdatendateien befinden sollen, müssen den Wert von **DB2INSTPROF** abfragen. Dabei handelt es sich um die folgenden Variablen:

- **DB2CLIINIPATH**
- **diagpath**
- **spm_log_path**

Alle Datenbankkonfigurationsparameter werden in einer Datei mit dem Namen SQLDBCONF gespeichert. Diese Dateien können nicht direkt editiert werden, sondern lediglich mithilfe einer bereitgestellten Anwendungsprogrammierschnittstelle (API) oder mit einem Tool, das diese API aufruft, geändert oder angezeigt werden.

In einer Umgebung mit partitionierten Datenbanken befindet sich diese Datei in einem gemeinsamen Dateisystem, sodass alle Datenbankpartitionsserver Zugriff auf dieselbe Datei haben. Die Konfiguration des Datenbankmanagers ist auf allen Datenbankpartitionsservern identisch.

Die meisten Parameter haben entweder Einfluss darauf, wie viel Systemressourcen einer einzelnen Datenbankmanagerinstanz zugeordnet werden, oder sie konfigurieren die Einrichtung des Datenbankmanagers und der verschiedenen Kommunikationssubsysteme nach umgebungsspezifischen Aspekten. Darüber hinaus gibt es noch weitere Parameter, die nur der Information dienen und deren Werte nicht geändert werden können. Alle diese Parameter sind allgemein gültig und unabhängig von einzelnen Datenbanken, die unter dieser Datenbankmanagerinstanz gespeichert sind.

Eine *Konfigurationsdatei der Datenbank* wird erstellt, wenn eine Datenbank erstellt wird. Sie befindet sich dort, wo sich die Datenbank befindet. Pro Datenbank gibt es eine Konfigurationsdatei. Die in ihr enthaltenen Parameter geben neben anderen Merkmalen die Menge der Ressourcen an, die der betreffenden Datenbank zugeordnet sind. Die Werte vieler dieser Parameter können zur Leistungsoptimierung und Kapazitätserhöhung geändert werden. Abhängig von der Art der Aktivitäten in einer bestimmten Datenbank können unterschiedliche Änderungen erforderlich sein.

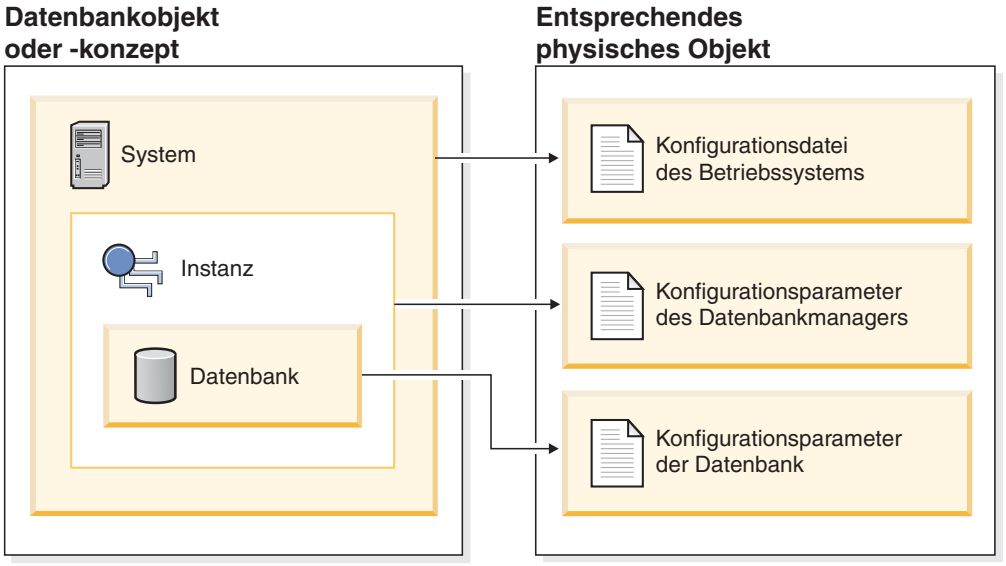


Abbildung 51. Beziehung zwischen Datenbankobjekten und Konfigurationsdateien

Konfigurieren des DB2-Datenbankmanagers mit Konfigurationsparametern

Der vom Datenbankmanager auf der Basis von Standardwerten für die Parameter zugeordnete Plattenspeicherplatz und der Hauptspeicher sind für Ihre Anforderungen möglicherweise ausreichend. In einigen Situationen wird jedoch die maximale Leistung bei Verwendung dieser Standardwerte möglicherweise nicht erreicht.

Informationen zu diesem Vorgang

Da sich die Standardwerte an Systemen orientieren, die über relativ kleine Hauptspeicherressourcen verfügen und als dedizierte Datenbankserver eingesetzt werden, müssen Sie diese Werte möglicherweise ändern, wenn Ihre Umgebung folgende Merkmale aufweist:

- Umfangreiche Datenbanken
- Große Anzahl von Verbindungen
- Hohe Leistungsanforderungen für eine bestimmte Anwendung
- Spezielle Abfrage-/Transaktionsauslastungen bzw. -typen

Jede Umgebung, die Transaktionen verarbeitet, hat einen oder mehrere spezifische, nur für sie geltende Aspekte. Diese Unterschiede können sich tief greifend auf die Leistung des Datenbankmanagers auswirken, wenn die Standardkonfiguration verwendet wird. Aus diesem Grund wird ausdrücklich empfohlen, die Konfiguration für die Umgebung zu optimieren.

Ein guter Ausgangspunkt für die Optimierung Ihrer Konfiguration ist die Verwendung des Konfigurationsadvisors bzw. des Befehls **AUTOCONFIGURE**. Diese Tools generieren Parameterwerte auf der Basis Ihrer Antworten auf Fragen zu Auslastungsmerkmalen.

Einige Konfigurationsparameter können auf AUTOMATIC gesetzt werden, sodass der Datenbankmanager diese Parameter automatisch so verwalten kann, dass sie den jeweils aktuellen Ressourcenbedarf berücksichtigen. Wenn Sie die Einstellung AAUTOMATIC eines Konfigurationsparameters inaktivieren und gleichzeitig die aktuelle interne Einstellung beibehalten wollen, verwenden Sie das Schlüsselwort **MANUAL** im Befehl **UPDATE DATABASE CONFIGURATION**. Wenn der Datenbankmanager den Wert dieser Parameter aktualisiert, wird über die Befehle **GET DB CFG SHOW DETAIL** und **GET DBM CFG SHOW DETAIL** der neue Wert angezeigt.

Parameter für eine einzelne Datenbank werden in einer Konfigurationsdatei mit dem Namen SQLDBCONF gespeichert. Diese Datei wird zusammen mit anderen Steuerdateien für die Datenbank im Verzeichnis SQL*nnnnn* gespeichert, wobei *nnnnn* eine Nummer ist, die bei der Erstellung der Datenbank zugeordnet wurde. Jede Datenbank hat eine eigene Konfigurationsdatei, und die meisten Parameter in der Datei geben an, wie viele Ressourcen der betreffenden Datenbank zugeordnet werden. Die Datei enthält außerdem beschreibende Informationen sowie Markierungen, die den Status der Datenbank angeben.

Achtung: Wenn Sie die Datei db2systm, SQLDBCON oder SQLDBCONF mit anderen als den vom Datenbankmanager vorgesehenen Methoden bearbeiten, wird die Datenbank möglicherweise unbrauchbar. Ändern Sie diese Dateien nicht mit anderen als den dokumentierten und vom Datenbankmanager unterstützten Methoden.

In einer Umgebung mit partitionierten Datenbanken ist für jede Datenbankpartition eine separate Datei SQLDBCONF vorhanden. Die Werte in der Datei SQLDBCONF können in den einzelnen Datenbankpartitionen gleich oder unterschiedlich sein. Für eine homogene Umgebung ist es allerdings empfehlenswert, in allen Datenbankpartitionen die gleichen Werte für die Datenbankkonfigurationsparameter zu verwenden. In der Regel könnte ein Katalogknoten andere Einstellungen für die Datenbankkonfigurationsparameter benötigen, während die anderen Datenpartitionen abhängig vom Systemtyp und anderen Informationen wiederum andere Werte haben.

Vorgehensweise

1. Aktualisieren Sie die Konfigurationsparameter.
   - Bei Verwendung des Befehlszeilenprozessors

     Befehle zum Ändern der Einstellungen können wie folgt eingegeben werden:

     Für Konfigurationsparameter des Datenbankmanagers:
     - **GET DATABASE MANAGER CONFIGURATION** (oder **GET DBM CFG**)
     - **UPDATE DATABASE MANAGER CONFIGURATION** (oder **UPDATE DBM CFG**)
     - **RESET DATABASE MANAGER CONFIGURATION** (oder **RESET DBM CFG**) zum Zurücksetzen *aller* Parameter des Datenbankmanagers auf ihre Standardwerte
     - **AUTOCONFIGURE**

     Für Konfigurationsparameter der Datenbank:
     - **GET DATABASE CONFIGURATION** (oder **GET DB CFG**)
     - **UPDATE DATABASE CONFIGURATION** (oder **UPDATE DB CFG**)
     - **RESET DATABASE CONFIGURATION** (oder **RESET DB CFG**) zum Zurücksetzen *aller* Datenbankparameter auf ihre Standardwerte
     - **AUTOCONFIGURE**
   - Bei Verwendung von Anwendungsprogrammierschnittstellen (APIs)

     APIs können in einer Anwendung oder einem Programm in einer Hostprogrammiersprache aufgerufen werden. Rufen Sie die folgenden DB2-APIs auf, um Konfigurationsparameter anzuzeigen oder zu aktualisieren:
     - db2AutoConfig - Auf den Konfigurationsadvisor zugreifen
     - db2CfgGet - Konfigurationsparameter für den Datenbankmanager oder die Datenbank abrufen
     - db2CfgSet - Konfigurationsparameter für den Datenbankmanager oder die Datenbank definieren
   - Bei Verwendung von Prozeduren der allgemeinen SQL-Anwendungsprogrammierschnittstelle (API)

     Sie können die allgemeinen Prozeduren der SQL-API in einer SQL-basierten Anwendung, über eine DB2-Befehlszeile oder über ein Befehlsscript aufrufen. Rufen Sie die folgenden Prozeduren auf, um Konfigurationsparameter anzuzeigen oder zu aktualisieren:
     - GET_CONFIG - Konfigurationsparameter für den Datenbankmanager oder die Datenbank abrufen
     - SET_CONFIG - Konfigurationsparameter für den Datenbankmanager oder die Datenbank definieren
   - Klicken Sie unter Verwendung von IBM Data Studio mit der rechten Maustaste auf die Instanz, um den Taskassistenten aufzurufen und die Konfigurationsparameter des Datenbankmanagers zu aktualisieren.
2. Zeigen Sie die aktualisierten Konfigurationswerte an.

   Für einige Konfigurationsparameter des Datenbankmanagers muss der Datenbankmanager gestoppt (**db2stop**) und anschließend erneut gestartet (**db2start**) werden, damit die neuen Parameterwerte in Kraft treten.

   Für einige Datenbankparameter werden Änderungen erst wirksam, wenn die Datenbank erneut aktiviert bzw. vom Offlinestatus in den Onlinestatus versetzt wird. Dazu müssen zunächst alle Anwendungen ihre Verbindung zur Datenbank trennen. (Wenn die Datenbank aktiviert war bzw. vom Offlinestatus in den Onlinestatus versetzt wurde, muss sie inaktiviert und anschließend erneut aktiviert werden.) Bei der Herstellung der ersten neuen Verbindung zur Datenbank werden die Änderungen wirksam.

Wenn Sie die Einstellung eines online konfigurierbaren Konfigurationsparameters des Datenbankmanagers ändern, während Sie mit einer Instanz verbunden sind, wendet der Befehl **UPDATE DBM CFG** die Änderung standardmäßig sofort an. Wenn die Änderung nicht sofort angewendet werden soll, verwenden Sie die Option **DEFERRED** im Befehl **UPDATE DBM CFG**.

Geben Sie folgende Befehle ein, um einen Konfigurationsparameter des Datenbankmanagers online zu ändern:

```
db2 attach to instance-name
db2 update dbm cfg using parameter-name value
db2 detach
```

Bei Clients werden Änderungen an den Konfigurationsparametern des Datenbankmanagers wirksam, wenn der Client das nächste Mal die Verbindung zu einem Server herstellt.

Wenn Sie einen online konfigurierbaren Datenbankkonfigurationsparameter ändern, während Sie mit einer Datenbank verbunden sind, wird die Änderung standardmäßig, soweit möglich, online angewendet. Beachten Sie jedoch, dass einige Parameteränderungen aufgrund der zusätzlichen Verarbeitungszeit, die durch die Speicherzuordnung anfällt, möglicherweise eine relativ lange Zeit benötigen, um in Kraft zu treten. Zur Änderung von Konfigurationsparametern online über den Befehlszeilenprozessor ist eine Verbindung zur Datenbank erforderlich. Geben Sie folgende Befehle ein, um einen Datenbankkonfigurationsparameter online zu ändern:

```
db2 connect to datenbankname
db2 update db cfg using parametername parameterwert
db2 connect reset
```

Jedem online konfigurierbaren Konfigurationsparameter ist eine *Weitergabeklasse* zugeordnet. Die Weitergabeklasse gibt an, wann Sie damit rechnen können, dass eine Änderung an dem Konfigurationsparameter in Kraft tritt. Es gibt vier Weitergabeklassen:

- **Sofort:** Gibt Parameterwerte an, die sich sofort beim Aufruf des Befehls oder der API ändern. Zum Beispiel hat der Parameter **diaglevel** die Weitergabeklasse 'Sofort'.
- **Anweisungsgrenzwert:** Gibt Parameter an, die sich bei Anweisungsgrenzen und anweisungsähnlichen Grenzen ändern. Wenn Sie zum Beispiel den Wert des Parameters **sortheap** ändern, verwenden alle neuen Anforderungen den neuen Wert.
- **Transaktionsgrenzwert:** Gibt Parameter an, die sich bei Transaktionsgrenzen ändern. Zum Beispiel wird ein neuer Wert für den Parameter **dl_expint** nach einer Anweisung COMMIT aktualisiert.
- **Verbindung**: Gibt Parameter an, die sich bei einer neuen Verbindung zur Datenbank ändern. Zum Beispiel wird ein neuer Wert für **dft_degree** wirksam, wenn neue Anwendungen eine Verbindung zur Datenbank herstellen.

Obwohl die neuen Parameterwerte möglicherweise nicht sofort wirksam sind, werden beim Anzeigen der Parametereinstellungen (mithilfe des Befehls **GET DATABASE MANAGER CONFIGURATION** bzw. **GET DATABASE CONFIGURATION**) stets die Werte der letzten Aktualisierung angezeigt. Wenn Sie die Klausel **SHOW DETAIL** in diesen Befehlen zum Anzeigen der Parametereinstellungen angeben, werden sowohl die Werte der letzten Aktualisierung als auch die im Hauptspeicher vorhandenen Werte angezeigt.

3. Führen Sie nach dem Aktualisieren von Datenbankkonfigurationsparametern einen Rebind für Anwendungen durch.

   Das Ändern einiger Konfigurationsparameter der Datenbank kann den Zugriffsplan beeinflussen, der vom SQL- und XQuery-Optimierungsprogramm ausgewählt wird. Ziehen Sie nach der Änderung eines solchen Parameters in

Betracht, einen Rebind für Ihre Anwendungen durchzuführen, um sicherzustellen, dass für Ihre SQL- und XQuery-Anweisungen der beste Zugriffsplan verwendet wird. Alle Parameter, die online modifiziert werden (z. B. durch den Befehl **UPDATE DATABASE CONFIGURATION IMMEDIATE**), veranlassen das SQL- und XQuery-Optimierungsprogramm, neue Zugriffspläne für neue Abfrageanweisungen auszuwählen. Allerdings wird der Abfrageanweisungscache nicht von vorhandenen Einträgen bereinigt. Zur Bereinigung des Inhalts des Abfragecache verwenden Sie die Anweisung FLUSH PACKAGE CACHE.

Anmerkung: Eine Reihe von Konfigurationsparametern (z. B. **health_mon**) verfügen laut Beschreibung in der Hilfe und anderer DB2-Dokumentation über die zulässigen Werte `Yes` oder `No` bzw. `On` oder `Off`. Zur Vermeidung von Unklarheiten sei hier angemerkt, dass `Yes` als äquivalent zu `On` und `No` als äquivalent zu `Off` anzusehen sind.

Zugehörige Informationen:

Konfigurationsparameter - Zusammenfassung

In den folgenden Tabellen sind die Parameter der Konfigurationsdateien des Datenbankmanagers und der Datenbank für Datenbankserver aufgeführt. Beachten Sie beim Ändern der Konfigurationsparameter des Datenbankmanagers und der Datenbank die detaillierten Informationen zu den einzelnen Parametern. Informationen zu spezifischen Betriebsumgebungen mit Standardwerten sind in jeder Parameterbeschreibung enthalten.

Übersicht über die Konfigurationsparameter des Datenbankmanagers

Für einige Konfigurationsparameter des Datenbankmanagers muss der Datenbankmanager gestoppt (**db2stop**) und erneut gestartet (**db2start**) werden, um die neuen Parameterwerte in Kraft zu setzen. Andere Parameter können online geändert werden. Diese werden als *online konfigurierbare Konfigurationsparameter* (Onl. kfg.) bezeichnet. Wenn Sie die Einstellung eines online konfigurierbaren Konfigurationsparameters des Datenbankmanagers ändern, während Sie mit einer Instanz verbunden sind, wird durch das Standardverhalten des Befehls **UPDATE DBM CFG** die Änderung unverzüglich angewendet. Wenn die Änderung nicht sofort angewendet werden soll, verwenden Sie die Option **DEFERRED** im Befehl **UPDATE DBM CFG**.

Die Spalte „Auto." in der folgenden Tabelle gibt an, ob der Parameter das Schlüsselwort **AUTOMATIC** im Befehl **UPDATE DBM CFG** unterstützt.

Bei der Aktualisierung eines Parameters auf den Wert AUTOMATIC ist es außerdem möglich, einen Anfangswert und das Schlüsselwort **AUTOMATIC** anzugeben. Beachten Sie hierbei, dass der Wert bei jedem Parameter eine andere Bedeutung haben kann und dass dieser in bestimmten Fällen nicht gilt. Vor der Angabe eines Werts sollten Sie die Dokumentation zu dem jeweiligen Parameter lesen, um festzustellen, was dieser Wert bedeutet. Im folgenden Beispiel wird **num_poolagents** auf den Wert `AUTOMATIC` aktualisiert, und der Datenbankmanager verwendet den Wert 20 als minimale Anzahl für inaktive Agenten, die in einem Pool zusammengefasst werden sollen:

```
db2 update dbm cfg using num_poolagents 20 automatic
```

Zur Inaktivierung der AUTOMATIC-Funktion kann der Parameter auf einem bestimmten Wert aktualisiert oder das Schlüsselwort **MANUAL** verwendet werden. Wenn ein Parameter auf `MANUAL` aktualisiert wird, wird der Parameter nicht mehr

automatisch festgelegt, sondern auf den aktuellen Wert gesetzt (wie in der Spalte `Aktueller Wert` der Ausgabe der Befehle **GET DBM CFG SHOW DETAIL** und **GET DB CFG SHOW DETAIL** angezeigt).

Wenn eine Datenbank mithilfe des Befehls **CREATE DATABASE** oder der Anwendungsprogrammierschnittstelle sqlecrea erstellt wurde, wird standardmäßig der Konfigurationsadvisor ausgeführt, um die Datenbankkonfigurationsparameter mit automatisch berechneten Werten zu aktualisieren. Wenn eine Datenbank mithilfe des Befehls **CREATE DATABASE** mit der hinzugefügten Klausel AUTOCONFIGURE APPLY NONE erstellt oder wenn durch die Anwendungsprogrammierschnittstelle sqlecrea angegeben wurde, dass der Konfigurationsadvisor nicht ausgeführt werden soll, werden für die Konfigurationsparameter die Standardwerte festgelegt.

In der Spalte „Leistungsrelevanz" (Leist.-Relev.) ist vermerkt, in welchem relativen Ausmaß sich jeder Parameter in Bezug auf die Systemleistung auswirken kann. Diese Spalte kann allerdings nicht für alle Arten von Umgebungen gültige Angaben enthalten. Sie sollten die Informationen als allgemeine Hinweise betrachten.

- **Hoch:** Gibt an, dass ein Parameter wesentliche Auswirkungen auf die Leistung haben kann. Die Werte für diese Parameter müssen sehr eingehend überlegt werden. In einigen Fällen kann das bedeuten, dass Sie die vorgegebenen Standardwerte übernehmen sollten.
- **Mittel:** Gibt an, dass der Parameter einige Auswirkungen auf die Leistung haben kann. Sie sollten von Ihrer spezifischen Umgebung und Ihren Anforderungen ausgehend entscheiden, wie viel Aufwand in die Optimierung dieser Parameter investiert werden sollte.
- **Niedrig:** Gibt an, dass der Parameter weniger allgemeine bzw. weniger bedeutende Auswirkungen auf die Leistung hat.
- **Keine:** Gibt an, dass der Parameter keine direkten Auswirkungen auf die Leistung hat. Da diese Parameter nicht leistungsrelevant sind, müssen sie nicht optimiert werden. Sie können jedoch in anderer Hinsicht für die Systemkonfiguration von Bedeutung sein, zum Beispiel zur Aktivierung der Kommunikationsunterstützung.

Die Spalten „Token", „Tokenwert" und „Datentyp" stellen Informationen bereit, die Sie beim Aufrufen der API db2CfgGet oder db2CfgSet benötigen. Diese Informationen enthalten Kennungen für Konfigurationsparameter, Einträge für das Element *token* in der Datenstruktur db2CfgParam und Datentypen für Werte, die an die Struktur übergeben werden.

Tabelle 125. Konfigurierbare Konfigurationsparameter des Datenbankmanagers

Parameter	Onl. kfg.	Auto.	Leist.- Relev.	Token	Token-wert	Datentyp	Zusätzliche Informationen
agent_stack_sz	Nein	Nein	Niedrig	SQLF_KTN_AGENT_STACK_SZ	61	Uint16	„agent_stack_sz - Größe des Agentenstacks" auf Seite 751
agentpri	Nein	Nein	Hoch	SQLF_KTN_AGENTPRI	26	Sint16	„agentpri - Agentenpriorität" auf Seite 752
alt_diagpath	Ja	Nein	Keine	SQLF_KTN_ALT_DIAGPATH SQLF_KTN_ALT_DIAGPATH_FULL	941	char [] (Zeichenfolge)	„alt_diagpath - Alternativer Verzeichnispfad für Diagnosedaten" auf Seite 754
alternate_auth_enc [6]	Nein	Nein	Niedrig	SQLF_KTN_ALTERNATE_AUTH_ENC	938	Uint16	„alternate_auth_enc - Alternativer Verschlüsselungsalgorithmus für ankommende Verbindungen auf dem Server (Konfigurationsparameter)" auf Seite 756
aslheapsz	Nein	Nein	Hoch	SQLF_KTN_ASLHEAPSZ	15	Uint32	„aslheapsz - Zwischenspeichergröße für Anwendungsunterstützungsebene" auf Seite 757
audit_buf_sz	Nein	Nein	Hoch	SQLF_KTN_AUDIT_BUF_SZ	312	Sint32	„audit_buf_sz - Prüfpuffergröße" auf Seite 759
authentication	Nein	Nein	Niedrig	SQLF_KTN_AUTHENTICATION	78	Uint16	„authentication - Authentifizierungstyp" auf Seite 759
catalog_noauth	Ja	Nein	Keine	SQLF_KTN_CATALOG_NOAUTH	314	Uint16	„catalog_noauth - Katalogisieren ohne Berechtigung zulässig" auf Seite 765

Tabelle 125. Konfigurierbare Konfigurationsparameter des Datenbankmanagers (Forts.)

Parameter	Onl. kfg.	Auto.	Leist.- Relev.	Token	Token-wert	Datentyp	Zusätzliche Informationen
cf_diaglevel	Nein	Nein	Keine	SQLF_KTN_CF_DIAGLEVEL	968	Uint16	„CF_diaglevel - Aufzeichnungsebene bei Fehlerdiagnose (Konfigurationsparameter für CF)" auf Seite 761
cf_diagpath	Nein	Nein	Keine	SQLF_KTN_CF_DIAGPATH SQLF_KTN_CF_DIAGPATH_ FULL	969	char(215)	„CF_diagpath - Verzeichnispfad für Diagnosedaten für CF (Konfigurationsparameter)" auf Seite 761
cf_mem_sz	Nein	Ja	Hoch	SQLF_KTN_CF_MEM_SZ	960	Uint32	„CF_mem_sz - CF-Speicher (Konfigurationsparameter)" auf Seite 762
cf_num_conns	Ja	Ja	Hoch	SQLF_KTN_CF_NUM_CONNS	966	Uint32	„cf_num_conns - Anzahl der CF-Verbindungen pro Member in CF (Konfigurationsparameter)" auf Seite 763
cf_num_workers	Nein	Ja	Hoch	SQLF_KTN_CF_NUM_WORKERS	961	Uint32	„CF_num_workers - Anzahl Worker-Threads (Konfigurationsparameter)" auf Seite 764
clnt_krb_plugin	Nein	Nein	Keine	SQLF_KTN_CLNT_KRB_ PLUGIN	812	char(33)	„clnt_krb_plugin - Client-Kerberos-Plug-in" auf Seite 766
clnt_pw_plugin	Nein	Nein	Keine	SQLF_KTN_CLNT_PW_ PLUGIN	811	char(33)	„clnt_pw_plugin - Client-Plug-in für Benutzer-ID und Kennwort" auf Seite 766
cluster_mgr	Nein	Nein	Keine	SQLF_KTN_CLUSTER_MGR	920	char(262)	„cluster_mgr - Name des Cluster-Managers" auf Seite 767
comm_bandwidth	Ja	Nein	Mittel	SQLF_KTN_COMM_ BANDWIDTH	307	float	„comm_bandwidth - Kommunikationsbandbreite" auf Seite 767
comm_exit_list	Nein	Nein	Niedrig	SQLF_KTN_COMM_EXIT_LIST	10121	char(129)	„comm_exit_list - Liste der Bibliotheken für Kommunikationspufferexits" auf Seite 768
conn_elapse	Ja	Nein	Mittel	SQLF_KTN_CONN_ELAPSE	508	Uint16	„conn_elapse - Antwortzeit für Verbindung" auf Seite 768
cpuspeed	Ja	Nein	Hoch	SQLF_KTN_CPUSPEED	42	float	„cpuspeed - CPU-Geschwindigkeit" auf Seite 769
dft_account_str	Ja	Nein	Keine	SQLF_KTN_DFT_ ACCOUNT_STR	28	char(25)	„dft_account_str - Standardzeichenfolge für Abrechnung" auf Seite 771
dft_monswitches • **dft_mon_bufpool** • **dft_mon_lock** • **dft_mon_sort** • **dft_mon_stmt** • **dft_mon_table** • **dft_mon_ timestamp** • **dft_mon_uow**	Ja	Nein	Mittel	SQLF_KTN_DFT_ MONSWITCHES[2] • SQLF_KTN_DFT_MON_ BUFPOOL • SQLF_KTN_DFT_MON_LOCK • SQLF_KTN_DFT_MON_SORT • SQLF_KTN_DFT_MON_STMT • SQLF_KTN_DFT_MON_ TABLE • SQLF_KTN_DFT_MON_ TIMESTAMP • SQLF_KTN_DFT_MON_ UOW	29 • 33 • 34 • 35 • 31 • 32 • 36 • 30	Uint16 • Uint16 • Uint16 • Uint16 • Uint16 • Uint16 • Uint16 • Uint16	„dft_monswitches - Monitorschalter des Standarddatenbanksystems" auf Seite 771
dftdbpath	Ja	Nein	Keine	SQLF_KTN_DFTDBPATH	27	char(215)	„dftdbpath - Standarddatenbankpfad" auf Seite 773
diaglevel	Ja	Nein	Niedrig	SQLF_KTN_DIAGLEVEL	64	Uint16	„diaglevel - Aufzeichnungsebene bei Fehlerdiagnose" auf Seite 774
diagpath	Ja	Nein	Keine	SQLF_KTN_DIAGPATH SQLF_KTN_DIAGPATH_FULL	65	char(215)	„diagpath - Verzeichnispfad für Diagnosedaten" auf Seite 775
dir_cache	Nein	Nein	Mittel	SQLF_KTN_DIR_CACHE	40	Uint16	„dir_cache - Verzeichniscacheunterstützung" auf Seite 781
discover[3]	Nein	Nein	Mittel	SQLF_KTN_DISCOVER	304	Uint16	„discover - Discovery-Modus" auf Seite 782
discover_inst	Ja	Nein	Niedrig	SQLF_KTN_DISCOVER_INST	308	Uint16	„discover_inst - Discovery-Serverinstanz" auf Seite 783
fcm_num_buffers	Ja	Ja	Mittel	SQLF_KTN_FCM_NUM_ BUFFERS	503	Uint32	„fcm_num_buffers - Anzahl FCM-Puffer" auf Seite 783
fcm_num_ channels	Ja	Ja	Mittel	SQLF_KTN_FCM_NUM_ CHANNELS	902	Uint32	„fcm_num_channels - Anzahl FCM-Kanäle" auf Seite 784
fed_noauth	Ja	Nein	Keine	SQLF_KTN_FED_NOAUTH	806	Uint16	„fed_noauth - Authentifizierung bei Servern mit föderierten Datenbanken umgehen" auf Seite 785
federated	Ja	Nein	Mittel	SQLF_KTN_FEDERATED	604	Sint16	„federated - Unterstützung für Systeme föderierter Datenbanken" auf Seite 786
federated_async	Ja	Ja	Mittel	SQLF_KTN_FEDERATED_ ASYNC	849	Sint32	„federated_async - Maximale ATQs pro Abfrage (Konfigurationsparameter)" auf Seite 786
fenced_pool	Ja	Ja	Mittel	SQLF_KTN_FENCED_POOL	80	Sint32	„fenced_pool - Maximale Anzahl abgeschirmter Prozesse" auf Seite 787
group_plugin	Nein	Nein	Keine	SQLF_KTN_GROUP_PLUGIN	810	char(33)	„group_plugin - Gruppen-Plug-in" auf Seite 789
health_mon	Ja	Nein	Niedrig	SQLF_KTN_HEALTH_MON	804	Uint16	„health_mon - Überwachung mit dem Diagnosemonitor" auf Seite 789

Tabelle 125. Konfigurierbare Konfigurationsparameter des Datenbankmanagers (Forts.)

Parameter	Onl. kfg.	Auto.	Leist.- Relev.	Token	Token-wert	Datentyp	Zusätzliche Informationen
indexrec[4]	Ja	Nein	Mittel	SQLF_KTN_INDEXREC	20	Uint16	„indexrec - Zeitpunkt für Indexneuerstellung" auf Seite 790
instance_memory	Ja	Ja	Mittel	SQLF_KTN_INSTANCE_ MEMORY	803	Uint64	„instance_memory - Instanzspeicher" auf Seite 792
intra_parallel	Nein	Nein	Hoch	SQLF_KTN_INTRA_PARALLEL	306	Sint16	„intra_parallel - Partitionsinterne Parallelität aktivieren" auf Seite 795
java_heap_sz	Nein	Nein	Hoch	SQLF_KTN_JAVA_HEAP_SZ	310	Sint32	„java_heap_sz - Maximale Zwischenspeichergröße für Java-Interpreter" auf Seite 796
jdk_path	Nein	Nein	Keine	SQLF_KTN_JDK_PATH	311	char(255)	„jdk_path - Installationspfad für Software Developer's Kit für Java ()" auf Seite 797
keepfenced	Nein	Nein	Mittel	SQLF_KTN_KEEPFENCED	81	Uint16	„keepfenced - Abgeschirmten Prozess beibehalten" auf Seite 797
local_gssplugin	Nein	Nein	Keine	SQLF_KTN_LOCAL_ GSSPLUGIN	816	char(33)	„local_gssplugin - GSS-API-Plug-in für lokale Berechtigung auf Instanzebene" auf Seite 798
max_connections	Ja	Ja	Mittel	SQLF_KTN_MAX_ CONNECTIONS	802	Sint32	„max_connections - Maximale Anzahl von Clientverbindungen" auf Seite 799
max_connretries	Ja	Nein	Mittel	SQLF_KTN_MAX_ CONNRETRIES	509	Uint16	„max_connretries - Anzahl Wiederholungen für Verbindungsaufbau zum Knoten" auf Seite 800
max_coordagents	Ja	Ja	Mittel	SQLF_KTN_MAX_ COORDAGENTS	501	Sint32	„max_coordagents - Maximale Anzahl koordinierender Agenten" auf Seite 800
max_querydegree	Ja	Nein	Hoch	SQLF_KTN_MAX_ QUERYDEGREE	303	Sint32	„max_querydegree - max_querydegree - Maximaler Grad der Parallelität bei Abfragen" auf Seite 802
max_time_diff	Nein	Nein	Mittel	SQLF_KTN_MAX_TIME_DIFF	510	Uint16	„max_time_diff - Maximale Zeitdifferenz zwischen Membern ()" auf Seite 802
mon_heap_sz	Ja	Ja	Niedrig	SQLF_KTN_MON_HEAP_SZ	79	Uint16	„mon_heap_sz - Zwischenspeichergröße für Datenbanksystemmonitor" auf Seite 805
notifylevel	Ja	Nein	Niedrig	SQLF_KTN_NOTIFYLEVEL	605	Sint16	„notifylevel - Benachrichtigungsstufe" auf Seite 807
num_initagents	Nein	Nein	Mittel	SQLF_KTN_NUM_ INITAGENTS	500	Uint32	„num_initagents - Anfangswert für die Anzahl Agenten im Pool" auf Seite 808
num_initfenced	Nein	Nein	Mittel	SQLF_KTN_NUM_ INITFENCED	601	Sint32	„num_initfenced - Anfangswert für die Anzahl abgeschirmter Prozesse" auf Seite 809
num_poolagents	Ja	Ja	Hoch	SQLF_KTN_NUM_ POOLAGENTS	502	Sint32	„num_poolagents - Agentenpoolgröße" auf Seite 809
numdb	Nein	Nein	Niedrig	SQLF_KTN_NUMDB	6	Uint16	„numdb - Maximale Anzahl gleichzeitig aktiver Datenbanken einschließlich Host- und System i-Datenbanken" auf Seite 810
query_heap_sz	Nein	Nein	Mittel	SQLF_KTN_QUERY_HEAP_SZ	49	Sint32	„query_heap_sz - Größe des Abfragezwischenspeichers" auf Seite 812
resync_interval	Nein	Nein	Keine	SQLF_KTN_RESYNC_ INTERVAL	68	Uint16	„resync_interval - Intervall für Transaktionsresynchronisation" auf Seite 814
rqrioblk	Nein	Nein	Hoch	SQLF_KTN_RQRIOBLK	1	Uint16	„rqrioblk - E/A-Blockgröße für Clients" auf Seite 815
sheapthres	Nein	Nein	Hoch	SQLF_KTN_SHEAPTHRES	21	Uint32	„sheapthres - Schwellenwert für Sortierspeicher" auf Seite 815
spm_log_file_sz	Nein	Nein	Niedrig	SQLF_KTN_SPM_LOG_FILE_ SZ	90	Sint32	„spm_log_file_sz - Protokolldateigröße für SPM" auf Seite 817
spm_log_path	Nein	Nein	Mittel	SQLF_KTN_SPM_LOG_PATH	313	char(226)	„spm_log_path - Protokolldateipfad für SPM" auf Seite 818
spm_max_resync	Nein	Nein	Niedrig	SQLF_KTN_SPM_MAX_ RESYNC	91	Sint32	„spm_max_resync - Maximale Anzahl von SPM-Resynchronisationsagenten" auf Seite 819
spm_name	Nein	Nein	Keine	SQLF_KTN_SPM_NAME	92	char(8)	„spm_name - Name des Synchronisationspunktmanagers" auf Seite 819
srvcon_auth	Nein	Nein	Keine	SQLF_KTN_SRVCON_AUTH	815	Uint16	„srvcon_auth - Authentifizierungstyp für ankommende Verbindungen auf dem Server" auf Seite 819
srvcon_gssplugin_ list	Nein	Nein	Keine	SQLF_KTN_SRVCON_ GSSPLUGIN_ LIST	814	char(256)	„srvcon_gssplugin_list - Liste der GSS-API-Plug-ins für ankommende Verbindungen auf dem Server" auf Seite 820
srv_plugin_mode	Nein	Nein	Keine	SQLF_KTN_SRV_PLUGIN_ MODE	809	Uint16	„srv_plugin_mode - Server-Plug-in-Modus" auf Seite 821
srvcon_pw_plugin	Nein	Nein	Keine	SQLF_KTN_SRVCON_PW_ PLUGIN	813	char(33)	„srvcon_pw_plugin - Plug-in für Benutzer-ID/Kennwort für ankommende Verbindungen auf dem Server" auf Seite 820

Tabelle 125. Konfigurierbare Konfigurationsparameter des Datenbankmanagers (Forts.)

Parameter	Onl. kfg.	Auto.	Leist.- Relev.	Token	Token-wert	Datentyp	Zusätzliche Informationen
ssl_svr_keydb	Nein	Nein	Keine	SQLF_KTN_SSL_SVR_KEYDB	930	char(1023)	„ssl_svr_keydb - SSL-Schlüsseldateipfad für ankommende SSL-Verbindungen auf dem Server (Konfigurationsparameter)" auf Seite 823
ssl_svr_stash	Nein	Nein	Keine	SQLF_KTN_SSL_SVR_STASH	931	char(1023)	„ssl_svr_stash - SSL-Stashdateipfad für ankommende SSL-Verbindungen auf dem Server (Konfigurationsparameter)" auf Seite 824
ssl_svr_label	Nein	Nein	Keine	SQLF_KTN_SSL_SVR_LABEL	932	char(1023)	„ssl_svr_label - Kennsatz in der Schlüsseldatei für ankommende SSL-Verbindungen auf dem Server (Konfigurationsparameter)" auf Seite 824
ssl_svcename	Nein	Nein	Keine	SQLF_KTN_SSL_SVCENAME	933	char(14)	„ssl_svcename - SSL-Servicename (Konfigurationsparameter)" auf Seite 826
ssl_cipherspecs	Nein	Nein	Keine	SQLF_KTN_SSL_CIPHERSPECS	934	char(255)	„ssl_cipherspecs - Unterstützte Verschlüsselungsspezifikationen auf dem Server (Konfigurationsparameter)" auf Seite 821
ssl_versions	Nein	Nein	Keine	SQLF_KTN_SSL_VERSIONS	935	char(255)	„ssl_versions - Unterstützte SSL-Versionen auf dem Server (Konfigurationsparameter)" auf Seite 827
ssl_clnt_keydb	Nein	Nein	Keine	SQLF_KTN_SSL_CLNT_KEYDB	936	char(1023)	„ssl_clnt_keydb - SSL-Schlüsseldateipfad für abgehende SSL-Verbindungen auf dem Client (Konfigurationsparameter)" auf Seite 822
ssl_clnt_stash	Nein	Nein	Keine	SQLF_KTN_SSL_CLNT_STASH	937	char(1023)	„ssl_clnt_stash - SSL-Stashdateipfad für abgehende SSL-Verbindungen auf dem Client (Konfigurationsparameter)" auf Seite 823
start_stop_time	Ja	Nein	Niedrig	SQLF_KTN_START_STOP_ TIME	511	Uint16	„start_stop_time - Zeitlimit für DB2START und DB2STOP" auf Seite 825
svcename	Nein	Nein	Keine	SQLF_KTN_SVCENAME	24	char(14)	„svcename - TCP/IP-Servicename" auf Seite 827
sysadm_group	Nein	Nein	Keine	SQLF_KTN_SYSADM_ GROUP	39	char(128)	„sysadm_group - SYSADM-Gruppenname" auf Seite 828
sysctrl_group	Nein	Nein	Keine	SQLF_KTN_SYSCTRL_ GROUP	63	char(128)	„sysctrl_group - SYSCTRL-Gruppenname" auf Seite 829
sysmaint_group	Nein	Nein	Keine	SQLF_KTN_SYSMAINT_ GROUP	62	char(128)	„sysmaint_group - SYSMAINT-Gruppenname" auf Seite 829
sysmon_group	Nein	Nein	Keine	SQLF_KTN_SYSMON_ GROUP	808	char(128)	„sysmon_group - Berechtigungsgruppenname für Systemmonitor" auf Seite 830
tm_database	Nein	Nein	Keine	SQLF_KTN_TM_DATABASE	67	char(8)	„tm_database - Name für Transaktionsmanagerdatenbank" auf Seite 830
tp_mon_name	Nein	Nein	Keine	SQLF_KTN_TP_MON_NAME	66	char(19)	„tp_mon_name - Name des Transaktionsprozessormonitors" auf Seite 831
trust_allclnts[5]	Nein	Nein	Keine	SQLF_KTN_TRUST_ALLCLNTS	301	Uint16	„trust_allclnts - Alle Clients akzeptieren" auf Seite 832
trust_clntauth	Nein	Nein	Keine	SQLF_KTN_TRUST_ CLNTAUTH	302	Uint16	„trust_clntauth - Authentifizierung gesicherter Clients" auf Seite 833
util_impact_lim	Ja	Nein	Hoch	SQLF_KTN_UTIL_IMPACT_ LIM	807	Uint32	„util_impact_lim - Richtlinie für Instanzauslastungswirkung" auf Seite 834
wlm_dispatcher	Ja	Nein	Mittel	SQLF_KTN_WLM_ DISPATCHER	976	Uint16	„wlm_dispatcher - Workload-Management-Dispatcher" auf Seite 835
wlm_disp_concur	Ja	Nein	Niedrig	SQLF_KTN_WLM_DISP_ CONCUR	977	Sint16	„wlm_disp_concur - Gemeinsamer Threadzugriff für Workload-Manager-Dispatcher ()" auf Seite 836
wlm_disp_cpu_shares	Ja	Nein	Niedrig	SQLF_KTN_WLM_DISP_ CPU_SHARES	979	Uint16	„wlm_disp_cpu_shares - CPU-Anteile des Workload-Manager-Dispatchers ()" auf Seite 837
wlm_disp_min_util	Ja	Nein	Niedrig	SQLF_KTN_WLM_DISP_ MIN_UTIL	978	Uint16	„wlm_disp_min_util - Minimale CPU-Auslastung des Workload-Manager-Dispatchers (" auf Seite 837

Tabelle 125. Konfigurierbare Konfigurationsparameter des Datenbankmanagers (Forts.)

Parameter	Onl. kfg.	Auto.	Leist.- Relev.	Token	Token-wert	Datentyp	Zusätzliche Informationen

Anmerkung:

In den Headerdateien `sqlenv.h` und `sqlutil.h` finden Sie die gültigen Werte und die Definitionen, die von den Konfigurationsparametern verwendet werden.

1.

```
Bit 1 (xxxx xxx1): dft_mon_uow
Bit 2 (xxxx xx1x): dft_mon_stmt
Bit 3 (xxxx x1xx): dft_mon_table
Bit 4 (xxxx 1xxx): dft_mon_buffpool
Bit 5 (xxx1 xxxx): dft_mon_lock
Bit 6 (xx1x xxxx): dft_mon_sort
Bit 7 (x1xx xxxx): dft_mon_timestamp
```

2. Gültige Werte:

```
SQLF_DSCVR_KNOWN (1)
SQLF_DSCVR_SEARCH (2)
```

3. Gültige Werte:

```
SQLF_INX_REC_SYSTEM (0)
SQLF_INX_REC_REFERENCE (1)
SQLF_INX_REC_RESTART (2)  SQLF_INX_REC_RESTART_NO_REDO (3)
SQLF_INX_REC_ACCESS_NO_REDO (4)
```

4. Gültige Werte:

```
SQLF_TRUST_ALLCLNTS_NO (0)
SQLF_TRUST_ALLCLNTS_YES (1)
SQLF_TRUST_ALLCLNTS_DRDAONLY (2)
```

5. Gültige Werte:

```
SQL_ALTERNATE_AUTH_ENC_AES (0)
SQL_ALTERNATE_AUTH_ENC_AES_CMP (1)
SQL_ALTERNATE_AUTH_ENC_NOTSPEC (255)
```

Tabelle 126. Informative Konfigurationsparameter des Datenbankmanagers

Parameter	Token	Token-wert	Datentyp	Zusätzliche Informationen
nodetype[1]	SQLF_KTN_NODETYPE	100	Uint16	„nodetype - Knotentyp der Instanz ()" auf Seite 806
release	SQLF_KTN_RELEASE	101	Uint16	„release - Release-Level der Datenbankkonfigurationsdatei" auf Seite 813

Anmerkung:

In den Headerdateien `sqlenv.h` und `sqlutil.h` finden Sie die gültigen Werte und die Definitionen, die von den Konfigurationsparametern verwendet werden.

1. Gültige Werte:

```
SQLF_NT_STANDALONE (0)
SQLF_NT_SERVER (1)
SQLF_NT_REQUESTOR (2)
SQLF_NT_STAND_REQ (3)
SQLF_NT_MPP (4)
SQLF_NT_SATELLITE (5)
```

Übersicht über die Konfigurationsparameter der Datenbank

In der folgenden Tabelle sind die Parameter der Konfigurationsdatei für die Datenbank aufgeführt. Wenn Sie Konfigurationsparameter der Datenbank ändern möchten, lesen Sie die detaillierten Informationen zu den Parametern.

Für einige Datenbankkonfigurationsparameter werden Änderungen erst dann wirksam, wenn die Datenbank erneut aktiviert wird. Dazu müssen zunächst alle Anwendungen ihre Verbindung zur Datenbank trennen. (Wenn die Datenbank aktiviert war, muss sie inaktiviert und anschließend erneut aktiviert werden.) Die Änderungen werden wirksam, sobald das nächste Mal eine Verbindung zur Datenbank hergestellt wird. Andere Parameter können online geändert werden. Diese werden als *online konfigurierbare Konfigurationsparameter* bezeichnet.

Eine Beschreibung der Spalten „Auto.", „Leist.-Relev.", „Token", „Tokenwert" und „Datentyp" finden Sie im Abschnitt mit der Übersicht über die Konfigurationsparameter des Datenbankmanagers.

Die Spalte „Member Cfg.” gilt nur für eine DB2 pureScale-Umgebung. Alle Datenbankkonfigurationsparameter können global konfiguriert werden. Diese Spalte gibt zusätzlich an, ob ein Datenbankkonfigurationsparameter auch auf Member-Ebene konfigurierbar ist. Weitere Informationen zu den Datenbankkonfigurationsparametern, die auf Member-Ebene konfiguriert werden können, finden Sie in DB2 pureScale Feature-Datenbankkonfigurationsparameter.

Das Schlüsselwort **AUTOMATIC** wird auch im Befehl **UPDATE DB CFG** unterstützt. Im folgenden Beispiel wird der Parameter **database_memory** auf den Wert AUTOMATIC aktualisiert, und der Datenbankmanager verwendet den Wert 20000 als Anfangswert, wenn weitere Änderungen an diesem Parameter vorgenommen werden:

```
db2 update db cfg using for sample using database_memory 20000 automatic
```

Ab Version 9.5 können Sie Datenbankkonfigurationsparameterwerte auf einigen oder allen Partitionen aktualisieren und zurücksetzen. Dazu müssen Sie nicht den Befehl **db2_all** absetzen oder jede Partition einzeln aktualisieren oder zurücksetzen.

Wenn eine Datenbank mithilfe des Befehls **CREATE DATABASE** oder der Anwendungsprogrammierschnittstelle sqlecrea erstellt wurde, wird standardmäßig der Konfigurationsadvisor ausgeführt, um die Datenbankkonfigurationsparameter mit automatisch berechneten Werten zu aktualisieren. Wenn eine Datenbank mithilfe des Befehls **CREATE DATABASE** mit der hinzugefügten Klausel AUTOCONFIGURE APPLY NONE erstellt oder wenn durch die Anwendungsprogrammierschnittstelle sqlecrea angegeben wurde, dass der Konfigurationsadvisor nicht ausgeführt werden soll, werden für die Konfigurationsparameter die Standardwerte festgelegt.

Tabelle 127. Konfigurierbare Konfigurationsparameter für die Datenbank

Parameter	Onl. kfg.	Auto.	Member Cfg.	Leist.-Relev.	Token	Token-wert	Daten-typ	Zusätzliche Informationen
alt_collate	Nein	Nein	Nein	Keine	SQLF_DBTN_ALT_COLLATE	809	Uint32	„alt_collate - Alternative Sortierfolge” auf Seite 838
applheapsz	Ja	Ja	Ja	Mittel	SQLF_DBTN_APPLHEAPSZ	51	Uint16	„applheapsz - Zwischenspeichergröße für Anwendungen” auf Seite 842
appl_memory	Ja	Ja	Ja	Mittel	SQLF_DBTN_APPL_MEMORY	904	Uint64	„appl_memory - Anwendungsspeicher (Konfigurationsparameter)” auf Seite 841
archretrydelay	Ja	Nein	Nein	Keine	SQLF_DBTN_ ARCHRETRYDELAY	828	Uint16	„archretrydelay - Wiederholungsintervall für Archivierung bei Fehler” auf Seite 843

Tabelle 127. Konfigurierbare Konfigurationsparameter für die Datenbank (Forts.)

Parameter	Onl. kfg.	Auto.	Member Cfg.	Leist.-Relev.	Token	Token-wert	Daten-typ	Zusätzliche Informationen
• **auto_ maint** • **auto_ db_backup** • **auto_ tbl_maint** • **auto_ runstats** • **auto_ stats_prof** • **auto_ stmt_stats** • **auto_ stats_views** • **auto_ prof_upd** • **auto_ reorg**	Ja	Nein	Nein	Mittel	• SQLF_DBTN_AUTO_MAINT • SQLF_DBTN_AUTO_DB_ BACKUP • SQLF_DBTN_AUTO_TBL_ MAINT • SQLF_DBTN_AUTO_ RUNSTATS • SQLF_DBTN_AUTO_STATS_ PROF • SQLF_DBTN_AUTO_STMT_ STATS • SQLF_DBTN_AUTO_STATS_ VIEWS • SQLF_DBTN_AUTO_PROF_ UPD • SQLF_DBTN_AUTO_REORG	• 831 • 833 • 835 • 837 • 839 • 905 • 980 • 844 • 841	Uint32	„auto_maint - Automatische Verwaltung" auf Seite 844
auto_del_ rec_obj	Ja	Nein	Nein	Mittel	SQLF_DBTN_AUTO_DEL_ REC_OBJ	912	Uint16	„auto_del_rec_obj - Automatisches Löschen von Recovery-Objekten (Konfigurationsparameter)" auf Seite 844
autorestart	Ja	Nein	Nein	Niedrig	SQLF_DBTN_AUTO_RESTART	25	Uint16	„autorestart - Automatischer Neustart aktiviert" auf Seite 848
auto_reval	Ja	Nein	Ja	Mittel	SQLF_DBTN_AUTO_REVAL	920	Uint16	„auto_reval - Automatische Reaktivierung und Inaktivierung (Konfigurationsparameter)" auf Seite 847
avg_appls	Ja	Ja	Ja	Hoch	SQLF_DBTN_AVG_APPLS	47	Uint16	„avg_appls - Durchschnittliche Anzahl aktiver Anwendungen" auf Seite 849
blk_log_dsk_ful	Ja	Nein	Ja	Keine	SQLF_DBTN_BLK_LOG_DSK_ FUL	804	Uint16	„blk_log_dsk_ful - Bei voller Protokollplatte blockieren" auf Seite 850
blocknonlogged	Ja	Nein	Ja	Niedrig	SQLF_DBTN_ BLOCKNONLOGGED	940	Uint16	„blocknonlogged - Erstellung von Tabellen blockieren, die nicht protokollierte Aktivitäten zulassen" auf Seite 851
catalogcache_sz	Ja	Nein	Ja	Mittel	SQLF_DBTN_ CATALOGCACHE_SZ	56	Uint32	„catalogcache_sz - Katalogcachegröße" auf Seite 856
chngpgs_thresh	Nein	Nein	Ja	Hoch	SQLF_DBTN_CHNGPGS_ THRESH	38	Uint16	„chngpgs_thresh - Schwellenwert für geänderte Seiten" auf Seite 858
connect_proc	Ja	Nein	Nein	Keine	SQLF_DBTN_CONNECT_PROC	954	char(257)	"cur_commit - Zurzeit festgeschriebene Daten (Konfigurationsparameter)"
cur_commit	Nein	Nein	Ja	Mittel	SQLF_DBTN_CUR_COMMIT	917	Uint32	„cur_commit - Zurzeit festgeschriebene Daten (Konfigurationsparameter)" auf Seite 770
database_memory	Ja	Ja	Ja	Mittel	SQLF_DBTN_DATABASE_ MEMORY	803	Uint64	„database_memory - Größe des gemeinsam genutzten Datenbankspeichers" auf Seite 862
dbheap	Ja	Ja	Ja	Mittel	SQLF_DBTN_DB_HEAP	58	Uint64	„dbheap - Zwischenspeicher für Datenbank" auf Seite 865

Tabelle 127. Konfigurierbare Konfigurationsparameter für die Datenbank (Forts.)

Parameter	Onl. kfg.	Auto.	Member Cfg.	Leist.-Relev.	Token	Token-wert	Daten-typ	Zusätzliche Informationen
db_mem_thresh	Ja	Nein	Ja	Niedrig	SQLF_DBTN_DB_MEM_ THRESH	849	Uint16	„db_mem_thresh - Schwellenwert für Datenbankspeicher" auf Seite 866
decflt_rounding	Nein	Nein	Nein	Keine	SQLF_DBTN_DECFLT_ ROUNDING	913	Uint16	„decflt_rounding - Rundungsmodus für dezimale Gleitkommawerte (Konfigurationsparameter)" auf Seite 868
dec_to_char_fmt	Ja	Ja	Ja	Mittel	SQLF_DBTN_DEC_TO_ CHAR_FMT	• 0: V95 • 1: NEW	Uint16	„dec_to_char_fmt - Funktion zur Konvertierung von Dezimalwerten in Zeichenwerte (Konfigurationsparameter)" auf Seite 867
dft_degree	Ja	Nein	Ja	Hoch	SQLF_DBTN_DFT_DEGREE	301	Sint32	„dft_degree - Grad der Parallelität" auf Seite 870
dft_extent_sz	Ja	Nein	Nein	Mittel	SQLF_DBTN_DFT_EXTENT_SZ	54	Uint32	„dft_extent_sz - Standardwert für EXTENTSIZE bei Tabellenbereichen" auf Seite 871
dft_loadrec_ses	Ja	Nein	Ja	Mittel	SQLF_DBTN_DFT_LOADREC_ SES	42	Sint16	„dft_loadrec_ses - Standardanzahl von Sitzungen für Recovery" auf Seite 872
dft_mttb_types	Nein	Nein	Nein	Keine	SQLF_DBTN_DFT_MTTB_ TYPES	843	Uint32	„dft_mttb_types - Standardtypen von verwalteten Tabellen zur Optimierung" auf Seite 872
dft_prefetch_sz	Ja	Ja	Nein	Mittel	SQLF_DBTN_DFT_PREFETCH_ SZ	40	Sint16	„dft_prefetch_sz - Standardwert für PREFETCHSIZE" auf Seite 873
dft_queryopt	Ja	Nein	Ja	Mittel	SQLF_DBTN_DFT_QUERYOPT	57	Sint32	„dft_queryopt - Standardabfrageoptimierungsklasse" auf Seite 874
dft_refresh_age	Nein	Nein	Nein	Mittel	SQLF_DBTN_DFT_REFRESH_ AGE	702	char(22)	„dft_refresh_age - Standardaktualisierungsalter" auf Seite 875
dft_schemas_dcc	Ja	Nein	Nein	Mittel	SQLF_DBTN_DFT_SCHEMAS_ DCC	10115	Uint16	„dft_schemas_dcc - Standarddatenerfassung für neue Schemas (Konfigurationsparameter)" auf Seite 875
discover_db	Ja	Nein	Nein	Mittel	SQLF_DBTN_DISCOVER	308	Uint16	„discover_db - Discovery-Unterstützung für diese Datenbank" auf Seite 877
dlchktime	Ja	Nein	Nein	Mittel	SQLF_DBTN_DLCHKTIME	9	Uint32	„dlchktime - Zeitintervall für Prüfung von Deadlocks" auf Seite 877
enable_xmlchar	Ja	Nein	Nein	Keine	SQLF_DBTN_ENABLE_ XMLCHAR	853	Uint32	„enable_xmlchar - Ermöglichen der Konvertierung in XML (Konfigurationsparameter)" auf Seite 878
failarchpath	Ja	Nein	Nein	Keine	SQLF_DBTN_FAILARCHPATH	826	char(243)	„failarchpath - Protokollarchivpfad für Funktionsübernahme" auf Seite 879
hadr_local_host	Nein	Nein	Nicht zutreffend	Keine	SQLF_DBTN_HADR_LOCAL_ HOST	811	char(256)	„hadr_local_host - Name des lokalen Hosts für HADR" auf Seite 880
hadr_local_svc	Nein	Nein	Nicht zutreffend	Keine	SQLF_DBTN_HADR_LOCAL_ SVC	812	char(41)	„hadr_local_svc - Lokaler HADR-Servicename" auf Seite 881

Tabelle 127. Konfigurierbare Konfigurationsparameter für die Datenbank (Forts.)

Parameter	Onl. kfg.	Auto.	Member Cfg.	Leist.-Relev.	Token	Token-wert	Daten-typ	Zusätzliche Informationen
hadr_peer_window	Nein	Nein	Nicht zutreffend	Niedrig (siehe Anmerkung 3)	SQLF_DBTN_HADR_PEER_WINDOW	914	Uint32	„hadr_peer_window - HADR-Peerfenster (Konfigurationsparameter)" auf Seite 882
hadr_remote_host	Nein	Nein	Nicht zutreffend	Keine	SQLF_DBTN_HADR_REMOTE_HOST	813	char(256)	„hadr_remote_host - Name des fernen HADR-Hosts" auf Seite 883
hadr_remote_inst	Nein	Nein	Nicht zutreffend	Keine	SQLF_DBTN_HADR_REMOTE_INST	815	char(9)	„hadr_remote_inst - HADR-Instanzname des fernen Servers" auf Seite 883
hadr_remote_svc	Nein	Nein	Nicht zutreffend	Keine	SQLF_DBTN_HADR_REMOTE_SVC	814	char(41)	„hadr_remote_svc - Ferner HADR-Servicename" auf Seite 883
hadr_replay_delay	Ja	Nein	Nicht zutreffend	Keine	SQLF_DBTN_HADR_REPLAY_DELAY	10119	Sint32	„hadr_replay_delay - HADR-Wiedergabeverzögerung (Konfigurationsparameter)" auf Seite 884
hadr_spool_limit	Ja	Nein	Nicht zutreffend	Keine	SQLF_DBTN_HADR_SPOOL_LIMIT	10112	Sint32	„hadr_spool_limit – Begrenzung für HADR-Protokollspooling (Konfigurationsparameter)" auf Seite 885
hadr_syncmode	Nein	Nein	Nicht zutreffend	Keine	SQLF_DBTN_HADR_SYNCMODE	817	Uint32	„hadr_syncmode - HADR-Synchronisationsmodus für Protokollierung im Peerstatus" auf Seite 886
hadr_target_list	Ja	Nein	Nicht zutreffend	Keine	SQLF_DBTN_HADR_TARGET_LIST	10114	char(890)	„hadr_target_list - Liste der HADR-Ziele (Datenbankkonfigurationsparameter)" auf Seite 888
hadr_timeout	Nein	Nein	Nicht zutreffend	Keine	SQLF_DBTN_HADR_TIMEOUT	816	Uint32	„hadr_timeout - HADR-Zeitlimitwert" auf Seite 890
indexrec[2]	Ja	Nein	Nein	Mittel	SQLF_DBTN_INDEXREC	30	Uint16	„indexrec - Zeitpunkt für Indexneuerstellung" auf Seite 790
locklist	Ja	Ja	Ja	Hoch bei Einfluss auf die Eskalation	SQLF_DBTN_LOCK_LIST	704	Uint64	„locklist - Maximaler Speicher für Sperrenliste" auf Seite 893
locktimeout	Nein	Nein	Ja	Mittel	SQLF_DBTN_LOCKTIMEOUT	34	Sint16	„locktimeout - Zeitlimit für Sperren" auf Seite 896
log_appl_info	Nein	Nein	Nicht zutreffend	Niedrig	SQLF_DBTN_LOG_APPL_INFO	10111	Uint32	„log_appl_info - Protokollsatz für Anwendungsinformationen (Datenbankkonfigurationsparameter)" auf Seite 897
log_ddl_stmts	Ja	Nein			SQLF_DBTN_LOG_DDL_STMTS	10110	Uint32	„log_ddl_stmts - DDL-Anweisungen protokollieren (Datenbankkonfigurationsparameter)" auf Seite 898
logarchcompr1	Ja	Nein	Nein	Keine	SQLF_DBTN_LOGARCHCOMPR1	10123	char(252)	„logarchcompr1 - Komprimierung für primäre archivierte Protokolldateien (Konfigurationsparameter)" auf Seite 898
logarchcompr2	Ja	Nein	Nein	Keine	SQLF_DBTN_LOGARCHCOMPR2	10124	char(252)	„logarchcompr2 - Komprimierung für sekundäre archivierte Protokolldateien (Konfigurationsparameter)" auf Seite 899
logarchmeth1	Ja	Nein	Nein	Keine	SQLF_DBTN_LOGARCHMETH1	822	char(252)	„logarchmeth1 - Primäre Protokollarchivierungsmethode" auf Seite 900

Tabelle 127. Konfigurierbare Konfigurationsparameter für die Datenbank (Forts.)

Parameter	Onl. kfg.	Auto.	Member Cfg.	Leist.-Relev.	Token	Token-wert	Daten-typ	Zusätzliche Informationen
logarchmeth2	Ja	Nein	Nein	Keine	SQLF_DBTN_ LOGARCHMETH2	823	char(252)	„logarchmeth2 - Sekundäre Protokollarchivierungsmethode" auf Seite 901
logarchopt1	Ja	Nein	Nein	Keine	SQLF_DBTN_LOGARCHOPT1	824	char(243)	„logarchopt1 - Optionen für primäres Protokollarchiv" auf Seite 903
logarchopt2	Ja	Nein	Nein	Keine	SQLF_DBTN_LOGARCHOPT2	825	char(243)	„logarchopt2 - Optionen für sekundäres Protokollarchiv" auf Seite 903
logbufsz	Nein	Nein	Ja	Hoch	SQLF_DBTN_LOGBUFSZ	33	Uint16	„logbufsz - Protokollpuffergröße" auf Seite 904
logfilsiz	Nein	Nein	Nein	Mittel	SQLF_DBTN_LOGFIL_SIZ	92	Uint32	„logfilsiz - Protokolldateigröße" auf Seite 905
logindexbuild	Ja	Nein	Ja	Keine	SQLF_DBTN_ LOGINDEXBUILD	818	Uint32	„logindexbuild - Erstellte Indexseiten protokollieren" auf Seite 906
logprimary	Nein	Nein	Nein	Mittel	SQLF_DBTN_LOGPRIMARY	16	Uint16	„logprimary - Anzahl primärer Protokolldateien" auf Seite 907
logsecond	Ja	Nein	Nein	Mittel	SQLF_DBTN_LOGSECOND	17	Uint16	„logsecond - Anzahl sekundärer Protokolldateien" auf Seite 909
max_log	Ja	Ja	Ja		SQLF_DBTN_MAX_LOG	807	Uint16	„max_log - Maximale Protokollgröße pro Transaktion" auf Seite 910
maxappls	Ja	Ja	Ja	Mittel	SQLF_DBTN_MAXAPPLS	6	Uint16	„maxappls - Maximale Anzahl aktiver Anwendungen" auf Seite 911
maxfilop	Ja	Nein	Ja	Mittel	SQLF_DBTN_MAXFILOP	3	Uint16	„maxfilop - Maximale Anzahl offener Datenbankdateien pro Datenbank" auf Seite 913
maxlocks	Ja	Ja	Ja	Hoch bei Einfluss auf die Eskalation	SQLF_DBTN_MAXLOCKS	15	Uint16	„maxlocks - Maximale Anzahl von Sperren vor Eskalation" auf Seite 913
min_dec_div_3	Nein	Nein	Nein	Hoch	SQLF_DBTN_MIN_DEC_DIV_3	605	Sint32	„min_dec_div_3 - 3 Kommastellen bei Dezimaldivision" auf Seite 915
mincommit	Ja	Nein	Ja	Hoch	SQLF_DBTN_MINCOMMIT	32	Uint16	„mincommit - Anzahl der Gruppencommits" auf Seite 916
mirrorlogpath	Nein	Nein	Nein	Niedrig	SQLF_DBTN_ MIRRORLOGPATH	806	char(242)	„mirrorlogpath - Pfad für Protokollspiegelung" auf Seite 918
mon_act_metrics	Ja	Nein	Ja	Mittel	SQLF_DBTN_MON_ ACT_METRICS	931	Uint16	„mon_act_metrics - Aktivitätsmesswerte überwachen (Konfigurationsparameter)" auf Seite 920
mon_deadlock	Ja	Nein	Ja	Mittel	SQLF_DBTN_MON_ DEADLOCK	934	Uint16	„mon_deadlock - Deadlocks überwachen (Konfigurationsparameter)" auf Seite 921
mon_locktimeout	Ja	Nein	Ja	Mittel	SQLF_DBTN_MON_ LOCKTIMEOUT	933	Uint16	„mon_locktimeout - Überschreitungen des Sperrzeitlimits überwachen (Konfigurationsparameter)" auf Seite 922

Tabelle 127. Konfigurierbare Konfigurationsparameter für die Datenbank (Forts.)

Parameter	Onl. kfg.	Auto.	Member Cfg.	Leist.-Relev.	Token	Token-wert	Daten-typ	Zusätzliche Informationen
mon_lockwait	Ja	Nein	Ja	Mittel	SQLF_DBTN_MON_ LOCKWAIT	935	Uint16	„mon_lockwait - Sperrwartestatus überwachen (Konfigurationsparameter)" auf Seite 923
mon_lw_thresh	Ja	Nein	Ja	Mittel	SQLF_DBTN_MON_ LW_THRESH	936	Uint32	„mon_lw_thresh - Schwellenwert für Sperrenwartestatus überwachen (Konfigurationsparameter)" auf Seite 924
mon_lck_msg_lvl	Ja	Nein	Ja	Keine	SQLF_DBTN_MON_ LCK_MSG_LVL	951	Uint16	„mon_lck_msg_lvl - Benachrichtigungen zu Sperrereignissen überwachen (Konfigurationsparameter)" auf Seite 924
mon_obj_metrics	Ja	Nein	Ja	Mittel	SQLF_DBTN_MON_ OBJ_METRICS	937	Uint16	„mon_obj_metrics - Datenobjektmesswerte überwachen (Konfigurationsparameter)" auf Seite 925
mon_pkglist_sz	Ja	Nein	Ja	Niedrig	SQLF_DBTN_MON_ PKGLIST_SZ	950	Uint32	„mon_pkglist_sz - Paketlistengröße überwachen (Konfigurationsparameter)" auf Seite 926
mon_req_metrics	Ja	Nein	Ja	Mittel	SQLF_DBTN_MON_ REQ_METRICS	930	Uint16	„mon_req_metrics - Anforderungsmesswerte überwachen (Konfigurationsparameter)" auf Seite 926
mon_uow_data	Ja	Nein	Ja	Mittel	SQLF_DBTN_MON_ UOW_DATA	932	Uint16	„mon_uow_data - UOW-Ereignisse überwachen (Konfigurationsparameter)" auf Seite 927
mon_uow_execlist	Ja	Nein	Ja	Mittel	SQLF_DBTN_MON_ UOW_EXECLIST	957	Uint16	„mon_uow_execlist - UOW-Ereignisüberwachung mit Liste ausführbarer Abschnitte (Konfigurationsparameter)" auf Seite 928
mon_uow_pkglist	Ja	Nein	Ja	Mittel	SQLF_DBTN_MON_ UOW_PKGLIST	956	Uint16	„mon_uow_pkglist - UOW-Ereignisüberwachung mit Paketliste (Konfigurationsparameter)" auf Seite 929
newlogpath	Nein	Nein	Nein	Niedrig	SQLF_DBTN_NEWLOGPATH	20	char(242)	„newlogpath - Datenbankprotokollpfad ändern" auf Seite 930
num_db_backups	Ja	Nein	Nein	Keine	SQLF_DBTN_NUM_ DB_BACKUPS	601	Uint16	„num_db_backups - Anzahl der Datenbank-Backups" auf Seite 932
num_freqvalues	Ja	Nein	Nein	Niedrig	SQLF_DBTN_NUM_ FREQVALUES	36	Uint16	„num_freqvalues - Anzahl der häufigsten Werte" auf Seite 932
num_iocleaners	Nein	Ja	Ja	Hoch	SQLF_DBTN_NUM_ IOCLEANERS	37	Uint16	„num_iocleaners - Anzahl asynchroner Seitenlöschfunktionen" auf Seite 934
num_ioservers	Nein	Ja	Ja	Hoch	SQLF_DBTN_NUM_ IOSERVERS	39	Uint16	„num_ioservers - Anzahl von E/A-Servern" auf Seite 935
num_log_span	Ja	Ja	Ja		SQLF_DBTN_NUM_ LOG_SPAN	808	Uint16	„num_log_span - Anzahl verwendeter Protokolldateien" auf Seite 936
num_quantiles	Ja	Nein	Ja	Niedrig	SQLF_DBTN_NUM_ QUANTILES	48	Uint16	„num_quantiles - Anzahl der Quantile für Spalten" auf Seite 937

Tabelle 127. Konfigurierbare Konfigurationsparameter für die Datenbank (Forts.)

Parameter	Onl. kfg.	Auto.	Member Cfg.	Leist.-Relev.	Token	Token-wert	Daten-typ	Zusätzliche Informationen
numarchretry	Ja	Nein	Ja	Keine	SQLF_DBTN_ NUMARCHRETRY	827	Uint16	„numarchretry - Anzahl der Wiederholungen bei Fehler" auf Seite 938
overflowlogpath	Ja	Nein	Nein	Mittel	SQLF_DBTN_ OVERFLOWLOGPATH	805	char(242)	„overflowlogpath - Überlaufprotokollpfad" auf Seite 939
pckcachesz	Ja	Ja	Ja	Hoch	SQLF_DBTN_PCKCACHE_SZ	505	Uint32	„pckcachesz - Größe des Paketcache" auf Seite 941
rec_his_retentn	Nein	Nein	Nein	Keine	SQLF_DBTN_REC_HIS_ RETENTN	43	Sint16	„rec_his_retentn - Aufbewahrungszeitraum für Recoveryprotokoll" auf Seite 944
section_actuals	Ja	Nein	Nein	Hoch	SQLF_DBTN_ SECTION_ACTUALS	952	Uint64	„section_actuals - Ist-Daten für Abschnitt (Konfigurationsparameter)" auf Seite 945
self_tuning_mem	Ja	Nein	Ja	Hoch	SQLF_DBTN_SELF_ TUNING_MEM	848	Uint16	„self_tuning_mem - Speicher mit automatischer Leistungsoptimierung" auf Seite 946
seqdetect	Ja	Nein	Nein	Hoch	SQLF_DBTN_SEQDETECT	41	Uint16	„seqdetect - Markierung für Sequenzerkennung und Vorauslesen (" auf Seite 948
sheapthres_shr	Ja	Ja	Ja	Hoch	SQLF_DBTN_SHEAPTHRES_ SHR	802	Uint32	„sheapthres_shr - Schwellenwert für Sortierspeicher für gemeinsame Sortiervorgänge" auf Seite 949
smtp_server	Ja	Nein	Ja	Keine	SQLF_DBTN_SMTP_SERVER	926	char [] (Zeichenfolge)	„smtp_server - SMTP-Server" auf Seite 950
softmax	Nein	Nein	Nein	Mittel	SQLF_DBTN_SOFTMAX	5	Uint16	„softmax - Recoverybereich und Intervall für bedingte Prüfpunkte" auf Seite 951
sortheap	Ja	Ja	Ja	Hoch	SQLF_DBTN_ SORT_HEAP	52	Uint32	„sortheap - Sortierspeichergröße" auf Seite 953
sql_ccflags	Ja	Nein	Ja	Keine	SQLF_DBTN_ SQL_CCFLAGS	927	char(1023)	„sql_ccflags - Markierungen für bedingte Kompilierung" auf Seite 955
stat_heap_sz	Ja	Ja	Ja	Niedrig	SQLF_DBTN_ STAT_HEAP_SZ	45	Uint32	„stat_heap_sz - Größe des Statistikzwischenspeichers" auf Seite 955
stmt_conc	Ja	Nein	Ja	Mittel	SQLF_DBTN_ STMT_CONC	919	Uint32	„stmt_conc - Anweisungskonzentrator (Konfigurationsparameter)" auf Seite 956
stmtheap	Ja	Ja	Ja	Mittel	SQLF_DBTN_ STMT_HEAP	821	Uint32	„stmtheap - Größe des Anweisungszwischenspeichers" auf Seite 957
systime_period _adj	Ja	Nein	Nein	Keine	SQLF_DBTN_SYSTIME_ PERIOD_ADJ	955	Uint16	„systime_period_adj - Temporalen Zeitraum für SYSTEM_TIME anpassen (Datenbankkonfigurationsparameter)" auf Seite 959
trackmod	Nein	Nein	Nein	Niedrig	SQLF_DBTN_TRACKMOD	703	Uint16	„trackmod - Geänderte Seiten protokollieren" auf Seite 960
tsm_mgmtclass	Ja	Nein	Ja	Keine	SQLF_DBTN_TSM_ MGMTCLASS	307	char(30)	„tsm_mgmtclass - Tivoli Storage Manager-Verwaltungsklasse" auf Seite 961
tsm_nodename	Ja	Nein	Ja	Keine	SQLF_DBTN_TSM_ NODENAME	306	char(64)	„tsm_nodename - Tivoli Storage Manager-Knotenname ()" auf Seite 961

Tabelle 127. Konfigurierbare Konfigurationsparameter für die Datenbank (Forts.)

Parameter	Onl. kfg.	Auto.	Member Cfg.	Leist.-Relev.	Token	Token-wert	Daten-typ	Zusätzliche Informationen
tsm_owner	Ja	Nein	Ja	Keine	SQLF_DBTN_TSM_ OWNER	305	char(64)	„tsm_owner - Tivoli Storage Manager-Eignername ()" auf Seite 962
tsm_password	Ja	Nein	Ja	Keine	SQLF_DBTN_TSM_ PASSWORD	501	char(64)	„tsm_password - Tivoli Storage Manager-Kennwort ()" auf Seite 962
util_heap_sz	Ja	Nein	Ja	Niedrig	SQLF_DBTN_UTIL_ HEAP_SZ	55	Uint32	„util_heap_sz - Zwischenspeichergröße für Dienstprogramme" auf Seite 963
vendoropt	Ja	Nein	Nein	Keine	SQLF_DBTN_VENDOROPT	829	char(242)	„vendoropt - Lieferantenoptionen" auf Seite 964<
wlm_collect_ int	Ja	Nein	Ja	Niedrig	SQLF_DBTN_WLM_ COLLECT_INT	907	Sint32	„wlm_collect_int - Workload-Management-Erfassungsintervall (Konfigurationsparameter)" auf Seite 965

Anmerkung:

In den Headerdateien `sqlenv.h` und `sqlutil.h` finden Sie die gültigen Werte und die Definitionen, die von den Konfigurationsparametern verwendet werden.

1. Die Bits des Tokens SQLF_DBTN_AUTONOMIC_SWITCHES geben die Standardeinstellungen für eine Reihe von Konfigurationsparametern zur automatischen Verwaltung an. Im Folgenden sehen Sie die einzelnen Bits, aus denen dieser zusammengesetzte Parameter besteht:

```
Default => Bit 1 on (xxxx xxxx xxxx xxx1): auto_maint
Bit 2 off (xxxx xxxx xxxx xx0x): auto_db_backup
Bit 3 on  (xxxx xxxx xxxx x1xx): auto_tbl_maint
Bit 4 on  (xxxx xxxx xxxx 1xxx): auto_runstats
Bit 5 off (xxxx xxxx xxx0 xxxx): auto_stats_prof
Bit 6 off (xxxx xxxx xx0x xxxx): auto_prof_upd
Bit 7 off (xxxx xxxx x0xx xxxx): auto_reorg
Bit 8 off (xxxx xxxx 0xxx xxxx): auto_storage
Bit 9 off (xxxx xxx0 xxxx xxxx): auto_stmt_stats
0 0 0 D

Maximum => Bit 1 on (xxxx xxxx xxxx xxx1): auto_maint
Bit 2 off (xxxx xxxx xxxx xx1x): auto_db_backup
Bit 3 on  (xxxx xxxx xxxx x1xx): auto_tbl_maint
Bit 4 on  (xxxx xxxx xxxx 1xxx): auto_runstats
Bit 5 off (xxxx xxxx xxx1 xxxx): auto_stats_prof
Bit 6 off (xxxx xxxx xx1x xxxx): auto_prof_upd
Bit 7 off (xxxx xxxx x1xx xxxx): auto_reorg
Bit 8 off (xxxx xxxx 1xxx xxxx): auto_storage
Bit 9 off (xxxx xxx1 xxxx xxxx): auto_stmt_stats
0 1 F F
```

2. Gültige Werte:

```
SQLF_INX_REC_SYSTEM (0)
SQLF_INX_REC_REFERENCE (1)
SQLF_INX_REC_RESTART (2)  SQLF_INX_REC_RESTART_NO_REDO (3)
SQLF_INX_REC_ACCESS_NO_REDO (4)
```

3. Wenn Sie den Parameter **hadr_peer_window** auf einen anderen Zeitwert als null setzen, kann die Primärdatenbank so aussehen, als ob sie bei Transaktionen blockiert, wenn sie sich im getrennten Peerzustand befindet, da sie auf eine Bestätigung von der Bereitschaftsdatenbank wartet, selbst wenn sie nicht mit der Bereitschaftsdatenbank verbunden ist.

Tabelle 128. Informative Konfigurationsparameter für die Datenbank

Parameter	Token	Token-wert	Datentyp	Zusätzliche Informationen
backup_pending	SQLF_DBTN_BACKUP_PENDING	112	Uint16	„backup_pending - Backup anstehend" auf Seite 850
codepage	SQLF_DBTN_CODEPAGE	101	Uint16	„codepage - Codepage für die Datenbank" auf Seite 859
codeset	SQLF_DBTN_CODESET	120	char(9)[1]	„codeset - Codierter Zeichensatz für die Datenbank" auf Seite 859
collate_info	SQLF_DBTN_COLLATE_INFO	44	char(260)	„collate_info - Informationen zur Sortierfolge" auf Seite 859

Tabelle 128. Informative Konfigurationsparameter für die Datenbank (Forts.)

Parameter	Token	Token-wert	Datentyp	Zusätzliche Informationen
country/region	SQLF_DBTN_COUNTRY	100	Uint16	„country/region - Gebietscode der Datenbank" auf Seite 861
database_consistent	SQLF_DBTN_CONSISTENT	111	Uint16	„database_consistent - Datenbank ist konsistent" auf Seite 861
database_level	SQLF_DBTN_DATABASE_LEVEL	124	Uint16	„database_level - Release-Level der Datenbank" auf Seite 862
hadr_db_role	SQLF_DBTN_HADR_DB_ROLE	810	Uint32	„hadr_db_role - Rolle der HADR-Datenbank" auf Seite 880
log_retain_status	SQLF_DBTN_LOG_RETAIN_STATUS	114	Uint16	„log_retain_status - Statusanzeiger für Beibehalten der Protokolle" auf Seite 898
loghead	SQLF_DBTN_LOGHEAD	105	char(12)	„loghead - Erste aktive Protokolldatei" auf Seite 906
logpath	SQLF_DBTN_LOGPATH	103	char(242)	„logpath - Pfad zu Protokolldateien" auf Seite 907
multipage_alloc	SQLF_DBTN_MULTIPAGE_ALLOC	506	Uint16	„multipage_alloc - Zuordnung aus mehreren Seiten bestehender Datei aktiv" auf Seite 930
numsegs	SQLF_DBTN_NUMSEGS	122	Uint16	„numsegs - Standardanzahl von SMS-Containern" auf Seite 939
pagesize	SQLF_DBTN_PAGESIZE	846	Uint32	„pagesize - Standardseitengröße für die Datenbank" auf Seite 941
release	SQLF_DBTN_RELEASE	102	Uint16	„release - Release-Level der Datenbankkonfigurationsdatei" auf Seite 813
restore_pending	SQLF_DBTN_RESTORE_PENDING	503	Uint16	„restore_pending - Restore anstehend" auf Seite 944
restrict_access	SQLF_DBTN_RESTRICT_ACCESS	852	Sint32	„restrict_access - Eingeschränkter Datenbankzugriff (Konfigurationsparameter)" auf Seite 945
rollfwd_pending	SQLF_DBTN_ROLLFWD_PENDING	113	Uint16	„rollfwd_pending - Aktualisierende Recovery anstehend" auf Seite 945
suspend_io	SQLF_DBTN_SUSPEND_IO	953	Uint16	„suspend_io - Status der Datenbank-E/A-Operationen (Konfigurationsparameter)" auf Seite 959
territory	SQLF_DBTN_TERRITORY	121	char(5)[2]	„territory - Datenbankgebiet" auf Seite 960
user_exit_status	SQLF_DBTN_USER_EXIT_STATUS	115	Uint16	„user_exit_status - Statusanzeiger für Benutzerexit" auf Seite 963

Anmerkung:

1. char(17) unter den Betriebssystemen HP-UX, Linux und und Solaris.
2. char(33) unter den Betriebssystemen HP-UX, Linux und und Solaris.

Übersicht über die Konfigurationsparameter für den DB2-Verwaltungsserver (DAS)

Wichtig: Der DB2-Verwaltungsserver (DAS) gilt in Version 9.7 als veraltet und wird möglicherweise in einem zukünftigen Release entfernt. Der DAS wird in DB2 pureScale-Umgebungen nicht unterstützt. Verwenden Sie Softwareprogramme, die das Secure Shell-Protokoll für die Fernverwaltung nutzen. Weitere Informationen hierzu finden Sie im Abschnitt „ DB2-Verwaltungsserver (DAS) gilt als veraltet" in http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0059276.html.

Tabelle 129. DAS-Konfigurationsparameter

Parameter	Parametertyp	Zusätzliche Informationen
authentication	Konfigurierbar	„authentication - DAS-Authentifizierungstyp" auf Seite 966
contact_host	Online konfigurierbar	„contact_host - Speicherposition der Liste mit Ansprechpartnern" auf Seite 966
das_codepage	Online konfigurierbar	„das_codepage - DAS-Codepage" auf Seite 967
das_territory	Online konfigurierbar	„das_territory - DAS-Gebiet" auf Seite 967
dasadm_group	Konfigurierbar	„dasadm_group - DASADM-Gruppenname" auf Seite 968
db2system	Online konfigurierbar	„db2system - Name des DB2-Serversystems" auf Seite 969
discover	Online konfigurierbar	„discover - DAS-Discovery-Modus" auf Seite 970
exec_exp_task	Konfigurierbar	„exec_exp_task - Verfallene Tasks ausführen" auf Seite 971
jdk_64_path	Online konfigurierbar	„jdk_64_path - Installationspfad für 64-Bit Software Developer's Kit für Java auf DAS" auf Seite 893
jdk_path	Online konfigurierbar	„jdk_path - Installationspfad für Software Developer's Kit für Java auf DAS" auf Seite 971
sched_enable	Konfigurierbar	„sched_enable - Schedulermodus" auf Seite 972
sched_userid	Informativ	„sched_userid - Benutzer-ID für Scheduler" auf Seite 972
smtp_server	Online konfigurierbar	„smtp_server - SMTP-Server" auf Seite 972
toolscat_db	Konfigurierbar	„toolscat_db - Toolskatalogdatenbank" auf Seite 973
toolscat_inst	Konfigurierbar	„toolscat_inst - Instanz der Toolskatalogdatenbank" auf Seite 973
toolscat_schema	Konfigurierbar	„toolscat_schema - Schema der Toolskatalogdatenbank" auf Seite 973

Übersicht über die Konfigurationsparameter des Aufnahmedienstprogramms

Tabelle 130. Konfigurationsparameter des Aufnahmedienstprogramms

Parameter	Parametertyp	Zusätzliche Informationen
commit_count	Konfigurierbar	„commit_count - Anzahl Commits (Konfigurationsparameter)" auf Seite 985
commit_period	Konfigurierbar	„commit_period - Commit-Zeitraum (Konfigurationsparameter)" auf Seite 986
num_flushers _per_partition	Konfigurierbar	„num_flushers_per_partition - Anzahl der Flusher pro Datenbankpartition (Konfigurationsparameter)" auf Seite 987
num_formatters	Konfigurierbar	„num_formatters - Anzahl der Formatierungsprogramme (Konfigurationsparameter)" auf Seite 988
pipe_timeout	Konfigurierbar	„pipe_timeout - Pipe-Zeitlimit (Konfigurationsparameter)" auf Seite 988
retry_count	Konfigurierbar	„retry_count - Wiederholungszähler (Konfigurationsparameter)" auf Seite 988
retry_period	Konfigurierbar	„retry_period - Wiederholungszeitlimit (Konfigurationsparameter)" auf Seite 989
shm_max_size	Konfigurierbar	„shm_max_size - Maximale Größe des gemeinsam genutzten Speichers (Konfigurationsparameter)" auf Seite 989

Abschnittsüberschriften für Konfigurationsparameter

Jede der Beschreibungen von Konfigurationsparametern enthält je nach Bedarf einige oder alle der folgenden Abschnittsüberschriften. In einigen Fällen schließen sie sich gegenseitig aus. Zum Beispiel müssen keine gültigen Werte angegeben werden, wenn der '[Bereich]' angegeben wird. In den meisten Fällen sind diese Überschriften selbsterklärend.

Tabelle 131. Beschreibung der Abschnittsüberschriften für Konfigurationsparameter

Abschnittsüberschrift	Beschreibung und mögliche Werte
Konfigurationstyp	Mögliche Werte: • Datenbankmanager • Datenbank • DB2-Verwaltungsserver

Tabelle 131. Beschreibung der Abschnittsüberschriften für Konfigurationsparameter (Forts.)

Abschnittsüberschrift	Beschreibung und mögliche Werte
Gilt für	Falls zutreffend, werden unter dieser Überschrift die Datenservertypen aufgelistet, für die der Konfigurationsparameter gilt. Mögliche Werte: • Client • Datenbankserver mit lokalen und fernen Clients • Datenbankserver mit lokalen Clients • DB2-Verwaltungsserver • OLAP-Funktionen • Partitionierten Datenbankserver mit lokalen und fernen Clients • Partitionierten Datenbankserver mit lokalen und fernen Clients, wenn die Föderation aktiviert ist. • Satellitendatenbankserver mit lokalen Clients
Parametertyp	Mögliche Werte: • Konfigurierbar (Der Datenbankmanager muss erneut gestartet werden, um die Änderungen in Kraft zu setzen.) • Online konfigurierbar (Der Konfigurationsparameter kann dynamisch online aktualisiert werden, ohne den Datenbankmanager erneut zu starten.) • Informativ (Werte dienen nur der Information und können nicht aktualisiert werden.) • Nach Member in einer DB2 pureScale-Umgebung konfigurierbar
Standardwert [Bereich]	Falls zutreffend, werden unter dieser Überschrift der Standardwert und die möglichen Wertebereiche, einschließlich Nullwerten (NULL) oder automatische Einstellungen (AUTOMATIC), angegeben. Wenn sich der Bereich je nach Plattform unterscheidet, werden die Werte nach Plattform bzw. Plattformtyp (z. B. 32-Bit-Plattformen oder 64-Bit-Plattformen) getrennt aufgeführt. Beachten Sie, dass der Standardwert in den meisten Fällen nicht als Teil des Bereichs angegeben wird.
Maßeinheit	Falls zutreffend, wird unter dieser Überschrift die Maßeinheit der Werte angegeben. Mögliche Werte: • Byte • Zähler • Megabyte pro Sekunde • Millisekunden • Minuten • Seiten (4 KB) • Prozent • Sekunden
Gültige Werte	Falls zutreffend, werden unter dieser Überschrift gültige Werte angegeben. Diese Überschrift schließt sich mit der Überschrift 'Standardwert [Bereich]' gegenseitig aus.
Beispiele	Falls zutreffend, werden unter dieser Überschrift Beispiele angegeben.
Weitergabeklasse	Falls zutreffend, sind die möglichen Werte folgende: • Sofort • Anweisungsgrenzwert • Transaktionsgrenzwert • Verbindung
Zuordnung	Falls zutreffend, wird unter dieser Überschrift angegeben, wann der Konfigurationsparameter vom Datenbankmanager zugeordnet wird.
Freigabe	Falls zutreffend, wird unter dieser Überschrift angegeben, wann der Konfigurationsparameter vom Datenbankmanager freigegeben wird.
Einschränkungen	Falls zutreffend, werden unter dieser Überschrift Einschränkungen aufgeführt, die für den Konfigurationsparameter gelten.
Begrenzungen	Falls zutreffend, werden unter dieser Überschrift Begrenzungen aufgeführt, die für den Konfigurationsparameter gelten.
Empfehlungen	Falls zutreffend, werden unter dieser Überschrift Empfehlungen gegeben, die für den Konfigurationsparameter gelten.
Hinweise	Falls zutreffend, werden unter dieser Überschrift Hinweise gegeben, die für den Konfigurationsparameter gelten.

Konfigurationsparameter mit Auswirkung auf die Anzahl von Agenten

Es gibt eine Reihe von Datenbankmanagerkonfigurationsparametern, die sich auf Datenbankagenten und deren Verwaltung beziehen.

Die folgenden Konfigurationsparameter des Datenbankmanagers legen fest, wie viele Datenbankagenten erstellt werden und wie sie verwaltet werden:

- Agentenpoolgröße (**num_poolagents**): Die Gesamtanzahl an inaktiven Agenten, die in einem Pool zusammengefasst werden sollen, die im System verfügbar gehalten werden. Der Standardwert für diesen Parameter ist `100, AUTOMATIC`.
- Anfangswert für die Anzahl von Agenten im Pool (**num_initagents**): Wenn der Datenbankmanager gestartet wird, wird ein Pool von Verarbeitungsagenten nach diesem Wert erstellt. Dadurch wird die Leistung für Erstabfragen erhöht. Die Verarbeitungsagenten beginnen alle als Agenten im Bereitschaftsmodus.
- Maximale Anzahl von Verbindungen (**max_connections**): Gibt die maximale Anzahl von Verbindungen zum Datenbankmanagersystem an, die in jeder Datenbankpartition zulässig sind.
- Maximale Anzahl koordinierender Agenten (**max_coordagents**): In Umgebungen mit partitionierten Datenbanken und Umgebungen mit aktivierter partitionsinterner Parallelität und aktiviertem Verbindungskonzentrator begrenzt dieser Wert die Anzahl koordinierender Agenten.

Konfigurationsparameter mit Einfluss auf die Abfrageoptimierung

Verschiedene Konfigurationsparameter wirken sich auf die Auswahl des Zugriffsplans durch den SQL- oder XQuery-Compiler aus. Viele von ihnen gelten für eine Umgebung mit Einzelpartitionsdatenbanken, während einige nur für eine Umgebung mit partitionierten Datenbanken gelten. In einer homogenen Umgebung mit partitionierten Datenbanken, in der identische Hardware eingesetzt wird, sollten die Werte, die für die einzelnen Parameter verwendet werden, für alle Datenbankpartitionen gleich sein.

Anmerkung: Wenn Sie einen Konfigurationsparameter dynamisch ändern, liest das Optimierungsprogramm die geänderten Parameterwerte möglicherweise nicht sofort, da vielleicht noch ältere Zugriffspläne im Paketcache vorhanden sind. Führen Sie zum Zurücksetzen des Paketcache die Anweisung FLUSH PACKAGE CACHE aus.

Wenn in einem föderierten System die Mehrzahl der Abfragen auf Kurznamen zugreift, sollten Sie die Art der Abfragen, die Sie senden, auswerten, bevor Sie Ihre Umgebung ändern. Zum Beispiel speichert der Pufferpool in einer föderierten Datenbank keine Seiten aus Datenquellen im Cache zwischen. Datenquellen sind die Datenbankverwaltungssysteme und Daten in einem föderierten System. Aus diesem Grund kann eine Erhöhung der Puffergröße nicht garantieren, dass das Optimierungsprogramm weitere Alternativen für Zugriffspläne in Betracht zieht, wenn es einen Zugriffsplan für Abfragen wählt, die Kurznamen enthalten. Allerdings stellt das Optimierungsprogramm möglicherweise fest, dass eine lokale Speicherung von Datenquellentabellen die Methode mit dem geringsten Aufwand oder ein erforderlicher Schritt für eine Sortieroperation ist. In diesem Fall kann eine Vergrößerung der verfügbaren Ressourcen die Leistung verbessern.

Die folgenden Konfigurationsparameter bzw. Faktoren wirken sich auf die Auswahl des Zugriffsplans durch den SQL- oder XQuery-Compiler aus:

- Die Größe der Pufferpools, die Sie angegeben haben, als Sie sie erstellt oder geändert haben

Bei der Auswahl des Zugriffsplans bezieht das Optimierungsprogramm den Ein-/Ausgabeaufwand für das Laden von Seiten von der Platte in den Pufferpool in die Kalkulation mit ein und schätzt die Anzahl der erforderlichen E/A-Operationen zur Erfüllung der Abfrage ab. Die Schätzung schließt eine Voraussage über die Nutzung des Pufferpools mit ein, da zum Lesen von Zeilen einer Seite, die sich bereits im Pufferpool befindet, keine weiteren physischen E/A-Operationen anfallen.

Das Optimierungsprogramm berücksichtigt den Wert der Spalte NPAGES in den Systemkatalogtabellen SYSCAT.BUFFERPOOLS und, in Umgebungen mit partitionierten Datenbanken, in den Systemkatalogtabellen SYSCAT.BUFFERPOOLDBPARTITIONS.

Der Ein-/Ausgabeaufwand für das Lesen der Tabellen kann sich auf folgende Bereiche auswirken:

– Wie zwei Tabellen verknüpft werden.
– Ob ein Index ohne Clustering zum Lesen der Daten verwendet wird.

- Standardgrad (**dft_degree**)

  Der Konfigurationsparameter **dft_degree** gibt die Parallelität durch Bereitstellen eines Standardwerts für das Sonderregister CURRENT DEGREE und die Bindeoption **DEGREE** an. Der Wert 1 bedeutet keine partitionsinterne Parallelität. Der Wert -1 bedeutet, dass das Optimierungsprogramm den Grad der partitionsinternen Parallelität je nach Anzahl der Prozessoren und Art der Abfrage festlegt.

  Anmerkung: Eine partitionsinterne Parallelverarbeitung findet nur statt, wenn Sie sie durch das Definieren des Konfigurationsparameters **intra_parallel** des Datenbankmanagers aktivieren.

- Standardabfrageoptimierungsklasse (**dft_queryopt**)

  Obwohl Sie beim Kompilieren von SQL- oder XQuery-Abfragen eine Abfrageoptimierungsklasse angeben können, können Sie außerdem eine Standardabfrageoptimierungsklasse definieren.

- Durchschnittliche Anzahl aktiver Anwendungen (**avg_appls**)

  Mithilfe des Parameters **avg_appls** versucht das Optimierungsprogramm zu ermitteln, wie viel vom Pufferpool während der Ausführung für den ausgewählten Zugriffsplan verfügbar ist. Höhere Werte für diesen Parameter können das Optimierungsprogramm dahin gehend beeinflussen, dass es Zugriffspläne auswählt, die mit dem Pufferpool etwas sparsamer umgehen. Wenn Sie den Wert 1 angeben, geht das Optimierungsprogramm davon aus, dass der gesamte Pufferpool für die Anwendung verfügbar ist.

- Zwischenspeicher für Sortierlisten (**sortheap**)

  Wenn die zu sortierenden Zeilen mehr als den im Sortierspeicher verfügbaren Speicherbereich in Anspruch nehmen, werden mehrere Sortierarbeitsgänge durchgeführt, wobei in jedem Arbeitsgang eine Untermenge der Gesamtmenge von Zeilen sortiert wird. Jeder Arbeitsgang des Sortiervorgangs wird in einer temporären Systemtabelle im Pufferpool gespeichert, die eventuell auf den Datenträger geschrieben wird. Wenn alle Arbeitsgänge des Sortiervorgangs abgeschlossen sind, werden die sortierten Untermengen zu einer einzigen sortierten Menge von Zeilen zusammengefügt. Eine Sortierung, bei der keine temporäre Systemtabelle zur Speicherung der Datenliste erforderlich ist, liefert stets die bessere Leistung und wird verwendet, fall möglich.

  Bei der Auswahl eines Zugriffsplans schätzt das Optimierungsprogramm den Aufwand der Sortieroperationen, einschließlich der Möglichkeiten, eine Sortierung mit einem einzelnen sequenziellen Zugriff zu lesen, folgendermaßen ab:

  – Abschätzen der Menge der zu sortierenden Daten

– Bestimmen mithilfe des Parameters **sortheap**, ob genügend Speicherbereich zur Verfügung steht, um eine Sortierung mit einem einzelnen sequenziellen Zugriff zu lesen.

- Maximaler Speicher für Sperrenliste (**locklist**) und maximale Anzahl von Sperren vor Eskalation (**maxlocks**)

  Wenn die Isolationsstufe RR (wiederholtes Lesen) verwendet wird, berücksichtigt das Optimierungsprogramm die Werte der Parameter **locklist** und **maxlocks**, um zu bestimmten, ob Sperren auf Zeilenebene möglicherweise durch Sperreneskalation in eine Sperre auf Tabellenebene umgewandelt werden. Wenn das Optimierungsprogramm eine Sperreneskalation für einen Tabellenzugriff voraussieht, wählt es für den Zugriffsplan eine Sperre auf Tabellenebene und vermeidet die zusätzliche Verarbeitungszeit, die mit einer Sperreneskalation während der Ausführung der Abfrage verbunden wäre.

- CPU-Geschwindigkeit (**cpuspeed**)

  Der Parameter für die CPU-Geschwindigkeit wird vom Optimierungsprogramm zur Abschätzung des Aufwands für bestimmte Operationen verwendet. Die Schätzwerte zum CPU-Aufwand und zu verschiedenen E/A-Aufwänden helfen bei der Auswahl des besten Zugriffsplans für eine Abfrage.

  Die CPU-Geschwindigkeit eines Systems kann die Auswahl des Zugriffsplans wesentlich beeinflussen. Dieser Konfigurationsparameter wird bei der Installation oder bei einem Upgrade der Datenbank automatisch auf geeigneten Wert gesetzt. Sie sollten diesen Parameter nicht anpassen, es sei denn, Sie wollen eine Produktionsumgebung auf einem Testsystem modellieren oder die Auswirkungen einer Änderung der Hardware testen. Mithilfe dieses Parameters können Sie eine andere Hardwareumgebung modellieren, um die Zugriffspläne zu ermitteln, die für die andere Umgebung ausgewählt würden. Wenn der Datenbankmanager den Wert dieses automatischen Konfigurationsparameters neu berechnen soll, setzen Sie den Parameter auf den Wert **-1**.

- Größe des Anweisungszwischenspeichers (**stmtheap**)

  Die Größe des Anweisungszwischenspeichers hat zwar keinen Einfluss darauf, welchen Zugriffspfad das Optimierungsprogramm auswählt, kann sich jedoch auf den Grad der Optimierung auswirken, der für komplexe SQL- oder XQuery-Anweisungen ausgeführt wird.

  Wenn der Wert für den Parameter **stmtheap** nicht groß genug ist, empfangen Sie möglicherweise eine Warnung, die angibt, dass nicht genügend Speicher zur Verarbeitung der Anweisung zur Verfügung steht. Zum Beispiel kann der SQLCODE +437 (SQLSTATE 01602) angeben, dass der Optimierungsgrad, der zur Kompilierung einer Anweisung verwendet wurde, geringer war als der Grad, den Sie angefordert haben.

- Kommunikationsbandbreite (**comm_bandwidth**)

  Die Kommunikationsbandbreite wird vom Optimierungsprogramm verwendet, um Zugriffspfade zu bestimmen. Das Optimierungsprogramm verwendet den Wert dieses Parameters, um den Aufwand zur Durchführung für bestimmte Operationen zwischen den Datenbankpartitionsservern in einer Umgebung mit partitionierten Datenbanken abzuschätzen.

- Zwischenspeichergröße für Anwendungen (**applheapsz**)

  Umfangreiche Schemata erfordern ausreichend Speicher im Anwendungszwischenspeicher.

Konfigurationsparameter mit Einfluss auf DB2 pureScale Feature

In einer DB2 pureScale-Umgebung werden die Datenbankkonfigurationsparameter entweder als *globale* Datenbankkonfigurationsparameter oder als *memberbezogene* Datenbankkonfigurationsparameter bezeichnet. Diese Unterscheidung ermöglicht es einer DB2 pureScale Feature-Instanz, von der globalen Konsistenz der Datenbankkonfiguration zu profitieren und gleichzeitig die Unterschiede zwischen den Membern zu berücksichtigen.

Globale Datenbankkonfigurationsparameter

Globale Datenbankkonfigurationsparameter nutzen gemeinsam einen konsistenten Wert für alle Member in einer DB2-Instanz. Eine Aktualisierung an einem Wert eines globalen Parameters gilt global für alle Member, unabhängig davon, von welchem Member die Aktualisierung ausgeführt wird. Die globale Konsistenz der Datenbankkonfiguration stellt sicher, dass alle Member in einer DB2-Instanz einheitlich auf dieselben Daten zugreifen und sie bearbeiten.

Memberbezogene Datenbankkonfigurationsparameter

Memberbezogene Datenbankkonfigurationsparameter können unterschiedliche Werte für die einzelnen Member in einer DB2-Instanz haben. Standardmäßig gilt eine Aktualisierung am Wert eines memberbezogenen Parameters global für alle Member, sofern diese Aktualisierung nicht nur für ein bestimmtes Member angegeben wird. Jedes Member in der DB2-Instanz kann eine Aktualisierung ausführen. Memberbezogene Datenbankkonfigurationsparameter im DB2 pureScale Feature sind für homogene Umgebungen konzipiert.

Für eine Datenbank, bei der sich die verfügbaren Ressourcen auf jedem Member unterscheiden, lässt sich durch eine Anpassung der Datenbankkonfiguration auf jedem Member die Datenbankleistung optimieren. Zum Beispiel können bei Unterschieden in den verfügbaren Speicherressourcen auf jedem Member durch individuelle Datenbankkonfigurationen auf den Membern Vorteile erzielt werden.

Eine Datenbank, bei der Member Anwendungen mit unterschiedlichen Funktionen zugeordnet sind, kann von angepassten Datenbankkonfigurationen auf den einzelnen Membern profitieren.

Aktualisieren der Datenbankkonfigurationsparameter in einer DB2 pureScale-Umgebung

Sie können Datenbankkonfigurationsparameter in einer DB2 pureScale-Instanz über den DB2-Befehlszeilenprozessor (CLP) oder mithilfe von DB2-Konfigurations-APIs aktualisieren. Die Verwendung der Klausel **MEMBER** ist optional und für memberbezogene Konfigurationsparameter gedacht.

Aktualisieren globaler Datenbankkonfigurationsparameter

Globale Datenbankkonfigurationsparameter werden in einer einzigen Datenbankkonfigurationsdatei gespeichert. Das aktualisierende Member ist für die Aktualisierung der Datenbankkonfigurationsdatei zuständig. Aktualisierungen an globalen Parametern schlagen nur fehl, wenn das aktualisierende Member nicht in die globale Konfigurationsdatei schreiben kann.

Anmerkung: Es ist kein Rollback erforderlich, wenn ein Member nicht in die Konfigurationsdatei schreiben kann, weil alle Member in der DB2 pureScale-Instanz sich in einem konsistenten Zustand befinden.

Wenn Sie versuchen, einen globalen Datenbankkonfigurationsparameter mit der Klausel **MEMBER** zu aktualisieren, empfangen Sie einen Fehler (SQL5125N).

Aktualisieren memberbezogener Datenbankkonfigurationsparameter

Member in einer DB2 pureScale-Instanz können mit unterschiedlichen Werten für die gleichen Konfigurationsparameter konfiguriert werden. Verwenden Sie die Klausel **MEMBER**, um eine Aktualisierung nur für ein Member in einer Instanz anzugeben. Wenn Sie die Klausel **MEMBER** nicht angeben, wird die Änderung auf alle Member in der Instanz angewendet.

Mit dem folgenden Befehl wird der Parameter **util_heap_sz** für MEMBER 2 auf den Wert 5000 aktualisiert:

```
UPDATE DATABASE CONFIGURATION FOR WSDB MEMBER 2 USING UTIL_HEAP_SZ 5000
```

Nur MEMBER 2 wird mit dem Wert 5000 für den Parameter **util_heap_sz** aktualisiert. Wenn Sie die Klausel **MEMBER** nicht angeben, wird der Parameter **util_heap_sz** für sämtliche Member in der DB2 pureScale auf den Wert 5000 aktualisiert.

Dieser Befehl kann über jedes Member in der DB2 pureScale-Instanz ausgeführt werden. Wenn MEMBER 2 ausfällt oder inaktiv wird, schlägt die Konfiguration nicht unbedingt fehl. Jedes Member in einer DB2 pureScale-Instanz kann in die Konfigurationsdatei eines anderen Members schreiben, sodass die Aktualisierung nur fehlschlägt, wenn das aktualisierende Member nicht erfolgreich in die Konfigurationsdatei von MEMBER 2 schreiben kann.

Anmerkung: Wenn die Klausel **MEMBER** nicht angegeben wird und das aktualisierende Member ohne Erfolg versucht, die Konfigurationsdatei für MEMBER 2 zu aktualisieren, wird die Aktualisierung in der gesamten DB2 pureScale-Instanz durch einen Rollback rückgängig gemacht. Wenn der Rollback ebenfalls fehlschlägt, können die Konfigurationsdateien in einem inkonsistenten Status zurückbleiben.

Bei Datenbankkonfigurationsparametern, die aktualisiert werden können, wenn die Datenbank online ist, werden keine zusätzlichen Maßnahmen ergriffen, um sicherzustellen, dass Werte in Cache und Speicher über eine DB2 pureScale-Instanz hinweg konsistenz sind.

Anmerkung: Die Werte für globale Konfigurationsparameter werden in der globalen Konfigurationsdatei im partitionsglobalen Verzeichnis gespeichert; die Werte für memberbezogene Konfigurationsparameter werden im memberspezifischen Verzeichnis gespeichert.

DB2 pureScale Feature-Konfigurationsparameter

Für IBM DB2 pureScale Feature sind verschiedene Konfigurationsparameter zur Unterstützung von Membern und der Cluster-Caching-Funktion (CF) vorhanden. Diese Datenbankkonfigurationsparameter werden in *globale* und *memberbezogene* Parameter untergliedert.

DB2 pureScale Feature-Konfigurationsparameter des Datenbankmanagers

In der folgenden Tabelle sind die Konfigurationsparameter des Datenbankmanagers für die Cluster-Caching-Funktion aufgeführt. Die Konfigurationsparameter des Datenbankmanagers gelten für die Konfiguration auf Instanzebene.

Tabelle 132. Zusammenfassung der DB2 pureScale Feature-Konfigurationsparameter des Datenbankmanagers

Parametername	Details
cf_diaglevel	Gibt die Diagnossestufe für Fehler an, die in der Datei `cfdiag.log` aufgezeichnet werden.
cf_diagpath	Gibt das Verzeichnis an, das die CF-Diagnoseprotokolldateien enthält, die auf der Basis des Parameters **cf_diaglevel** generiert werden.
cf_mem_size	Legt die Gesamtspeichergröße fest, die von der CF verwendet wird. Der Wert muss kleiner als die Größe des physischen Speichers auf dem CF-Host sein.
cf_num_conns	Legt die Anfangsgröße für den CF-Verbindungspool fest.
cf_num_workers	Legt fest, wie viele Worker-Threads vom CF-Server gestartet werden. Der Wert ist für gewöhnlich kleiner oder gleich der Anzahl der CPUs auf dem CF-Server.

DB2 pureScale Feature-Datenbankkonfigurationsparameter

In der folgenden Tabelle sind die Datenbankkonfigurationsparameter für die Cluster-Caching-Funktion aufgeführt. Die Datenbankkonfigurationsparameter gelten für die Konfiguration auf Datenbankebene.

Anmerkung: Der CF-Datenbankkonfigurationsparameter CF_db_mem_sz gilt auf einer datenbankweiten Ebene. Die übrigen CF-Datenbankkonfigurationsparameter gelten auf der Strukturebene und haben eine Obergrenze, die durch den Parameter 'CF_db_mem_sz' festgelegt wird. Weitere Informationen zur Beziehung zwischen den Datenbankkonfigurationsparametern finden Sie im Abschnitt zum Konfigurieren des Cluster-Caching-Funktion-Speichers für eine Datenbank.

Tabelle 133. Zusammenfassung der globalen DB2 pureScale Feature-Datenbankkonfigurationsparameter

Parametername	Details
cf_catchup_trgt	Gibt die Sollzeit in Minuten bis zum Abschluss der Angleichung (Catchup) an, durch die eine neu hinzugefügte bzw. erneut gestartete CF in den PEER-Status mit einer vorhandenen primären CF gebracht wird.
cf_db_mem_sz	Legt die Gesamtspeicherbegrenzung für eine CF für die aktuelle Datenbank fest.
cf_gbp_sz	Legt die Gesamtspeichergröße fest, die durch die CF für den Gruppenpufferpool für die aktuelle Datenbank zugeordnet wird.
cf_lock_sz	Legt die Gesamtspeichergröße fest, die durch die CF für die Sperrstruktur für die aktuelle Datenbank zugeordnet wird.
cf_sca_sz	Legt die Gesamtspeichergröße fest, die durch die CF für den gemeinsamen Kommunikationsbereich (SCA) für die aktuelle Datenbank zugeordnet wird.

Die folgenden DB2-Datenbankkonfigurationsparameter werden als *globale* Datenbankkonfigurationsparameter für DB2 pureScale Feature bezeichnet. Weitere Informationen zu globalen Datenbankkonfigurationsparametern finden Sie im Abschnitt "Datenbankkonfigurationsparameter für DB2 pureScale Feature".

Tabelle 134. Konfigurierbare globale Datenbankkonfigurationsparameter

Parameter	Zusätzliche Informationen
alt_collate	**alt_collate - Alternative Sortierfolge (Konfigurationsparameter)**
archretrydelay	**archretrydelay - Wiederholungsintervall für Archivierung bei Fehler (Konfigurationsparameter)**
auto_del_rec_obj	auto_del_rec_obj - Automatisches Löschen von Recovery-Objekten (Konfigurationsparameter)

Tabelle 134. Konfigurierbare globale Datenbankkonfigurationsparameter (Forts.)

Parameter	Zusätzliche Informationen
• auto_maint • auto_db_backup • auto_tbl_maint • auto_runstats • auto_stats_prof • auto_stmt_stats • auto_prof_upd • auto_reorg	auto_maint - Automatische Verwaltung (Konfigurationsparameter)
autorestart	autorestart - Automatischer Neustart aktiviert (Konfigurationsparameter)
decflt_rounding	decflt_rounding - Rundungsmodus für dezimale Gleitkommawerte (Konfigurationsparameter)
dft_extent_sz	dft_extent_sz - Standardwert für EXTENTSIZE bei Tabellenbereichen (Konfigurationsparameter)
dft_mttb_types	dft_mttb_types - Standardtypen von verwalteten Tabellen zur Optimierung (Konfigurationsparameter)
dft_prefetch_sz	dft_prefetch_sz - Standardwert für PREFETCHSIZE (Konfigurationsparameter)
dft_refresh_age	dft_refresh_age - Standardaktualisierungsalter (Konfigurationsparameter)
dft_sqlmathwarn	dft_sqlmathwarn - Bei arithmetischen Ausnahmebedingungen fortsetzen (Konfigurationsparameter)
discover_db	discover_db - Discovery-Unterstützung für diese Datenbank (Konfigurationsparameter)
dlchktime	dlchktime - Zeitintervall für Prüfung von Deadlocks (Konfigurationsparameter)
enable_xmlchar	enable_xmlchar - Ermöglichen der Konvertierung in XML (Konfigurationsparameter)
failarchpath	failarchpath - Protokollarchivpfad für Funktionsübernahme (Konfigurationsparameter)
indexrec	indexrec - Zeitpunkt für Indexneuerstellung (Konfigurationsparameter)
locktimeout	locktimeout - Zeitlimit für Sperren (Konfigurationsparameter)
logarchmeth1	logarchmeth1 - Primäre Protokollarchivierungsmethode (Konfigurationsparameter)
logarchmeth2	logarchmeth2 - Sekundäre Protokollarchivierungsmethode (Konfigurationsparameter)
logarchopt1	logarchopt1 - Optionen für primäres Protokollarchiv (Konfigurationsparameter)
logarchopt2	logarchopt2 - Optionen für sekundäres Protokollarchiv (Konfigurationsparameter)
logfilsiz	logfilsiz - Protokolldateigröße (Konfigurationsparameter)
logprimary	logprimary - Anzahl primärer Protokolldateien (Konfigurationsparameter)
logsecond	logsecond - Anzahl sekundärer Protokolldateien (Konfigurationsparameter)
min_dec_div_3	min_dec_div_3 - 3 Kommastellen bei Dezimaldivision (Konfigurationsparameter)
mirrorlogpath	mirrorlogpath - Pfad für Protokollspiegelung (Konfigurationsparameter)
mon_act_metrics	mon_act_metrics - Aktivitätsmesswerte überwachen (Konfigurationsparameter)
mon_deadlock	mon_deadlock - Deadlocks überwachen (Konfigurationsparameter)
mon_locktimeout	mon_locktimeout - Überschreitungen des Sperrzeitlimits überwachen (Konfigurationsparameter)
mon_lockwait	mon_lockwait - Sperrwartestatus überwachen (Konfigurationsparameter)

Tabelle 134. Konfigurierbare globale Datenbankkonfigurationsparameter (Forts.)

Parameter	Zusätzliche Informationen
mon_lw_thresh	mon_lw_thresh - Schwellenwert für Sperrenwartestatus überwachen (Konfigurationsparameter)
mon_obj_metrics	mon_obj_metrics - Datenobjektmesswerte überwachen (Konfigurationsparameter)
mon_req_metrics	mon_req_metrics - Anforderungsmesswerte überwachen (Konfigurationsparameter)
mon_uow_data	mon_uow_data - UOW-Ereignisse überwachen (Konfigurationsparameter)
newlogpath	newlogpath - Datenbankprotokollpfad ändern (Konfigurationsparameter)
num_db_backups	num_db_backups - Anzahl der Datenbank-Backups (Konfigurationsparameter)
num_freqvalues	num_freqvalues - Anzahl der häufigsten Werte (Konfigurationsparameter)
numarchretry	numarchretry - Anzahl der Wiederholungen bei Fehler (Konfigurationsparameter)
overflowlogpath	overflowlogpath - Überlaufprotokollpfad (Konfigurationsparameter)
rec_his_retentn	rec_his_retentn - Aufbewahrungszeitraum für Recoveryprotokoll (Konfigurationsparameter)
seqdetect	seqdetect - Markierung für Sequenzerkennung (Konfigurationsparameter)
softmax	softmax - Recoverybereich und Intervall für bedingte Prüfpunkte (Konfigurationsparameter)
trackmod	trackmod - Geänderte Seiten protokollieren (Konfigurationsparameter)
vendoropt	vendoropt - Lieferantenoptionen (Konfigurationsparameter)

Die folgenden DB2-Datenbankkonfigurationsparameter werden als *memberbezogene* Datenbankkonfigurationsparameter für DB2 pureScale Feature bezeichnet. Weitere Informationen zu memberbezogenen Datenbankkonfigurationsparametern finden Sie im Abschnitt "Datenbankkonfigurationsparameter für DB2 pureScale Feature".

Tabelle 135. Konfigurierbare memberbezogene Datenbankkonfigurationsparameter

Parameter	Zusätzliche Informationen
applheapsz	applheapsz - Zwischenspeichergröße für Anwendungen (Konfigurationsparameter)
appl_memory	appl_memory - Anwendungsspeicher (Konfigurationsparameter)
auto_reval	
avg_appls	avg_appls - Durchschnittliche Anzahl aktiver Anwendungen (Konfigurationsparameter)
blk_log_dsk_ful	blk_log_dsk_ful - Bei voller Protokollplatte blockieren (Konfigurationsparameter)
blocknonlogged	blocknonlogged - Erstellung von Tabellen blockieren, die nicht protokollierte Aktivitäten zulassen (Konfigurationsparameter)
catalogcache_sz	catalogcache_sz - Katalogcachegröße (Konfigurationsparameter)
chngpgs_thresh	chngpgs_thresh - Schwellenwert für geänderte Seiten (Konfigurationsparameter)
connect_proc	connect_proc - Name der Verbindungsprozedur (Datenbankkonfigurationsparameter)
cur_commit	cur_commit - Zurzeit festgeschriebene Daten (Konfigurationsparameter)
database_memory	database_memory - Größe des gemeinsam genutzten Datenbankspeichers (Konfigurationsparameter)
dbheap	dbheap - Zwischenspeicher für Datenbank (Konfigurationsparameter)
db_mem_thresh	db_mem_thresh - Schwellenwert für Datenbankspeicher (Konfigurationsparameter)
dec_to_char_fmt	dec_to_char_fmt - Funktion zur Konvertierung von Dezimalwerten in Zeichenwerte (Konfigurationsparameter)

Tabelle 135. Konfigurierbare memberbezogene Datenbankkonfigurationsparameter (Forts.)

Parameter	Zusätzliche Informationen
dft_degree	dft_degree - Grad der Parallelität (Konfigurationsparameter)
dft_loadrec_ses	dft_loadrec_ses - Standardanzahl von Sitzungen für Recovery (Konfigurationsparameter)
dft_queryopt	dft_queryopt - Standardabfrageoptimierungsklasse (Konfigurationsparameter)
discover_db	discover_db - Discovery-Unterstützung für diese Datenbank (Konfigurationsparameter)
hadr_local_host	hadr_local_host - Name des lokalen HADR-Hosts (Konfigurationsparameter)
hadr_local_svc	hadr_local_svc - Lokaler HADR-Servicename (Konfigurationsparameter)
hadr_peer_ window	hadr_peer_window - HADR-Peerfenster (Konfigurationsparameter)
hadr_remote_host	hadr_remote_host - Name des fernen HADR-Hosts (Konfigurationsparameter)
hadr_remote_inst	hadr_remote_inst - HADR-Instanzname des fernen Servers (Konfigurationsparameter)
hadr_remote_svc	hadr_remote_svc - Ferner HADR-Servicename (Konfigurationsparameter)
hadr_syncmode	hadr_syncmode - HADR-Synchronisationsmodus für Protokollierung im Peerstatus (Konfigurationsparameter)
hadr_timeout	hadr_timeout - HADR-Zeitlimitwert (Konfigurationsparameter)
locklist	locklist - Maximaler Speicher für Sperrenliste (Konfigurationsparameter)
locktimeout	locktimeout - Zeitlimit für Sperren (Konfigurationsparameter)
logbufsz	logbufsz - Protokollpuffergröße (Konfigurationsparameter)
logindexbuild	logindexbuild - Erstellte Indexseiten protokollieren (Konfigurationsparameter)
max_log	max_log - Maximale Protokollgröße pro Transaktion (Konfigurationsparameter)
maxappls	maxappls - Maximale Anzahl aktiver Anwendungen (Konfigurationsparameter)
maxfilop	maxfilop - Maximale Anzahl offener Datenbankdateien pro Anwendung (Konfigurationsparameter)
maxlocks	maxlocks - Maximale Anzahl von Sperren vor Eskalation (Konfigurationsparameter)
mincommit	mincommit - Anzahl der Gruppencommits (Konfigurationsparameter)
mon_act_metrics	mon_act_metrics - Aktivitätsmesswerte überwachen (Konfigurationsparameter)
mon_deadlock	mon_deadlock - Deadlocks überwachen (Konfigurationsparameter)
mon_locktimeout	mon_locktimeout - Überschreitungen des Sperrzeitlimits überwachen (Konfigurationsparameter)
mon_lockwait	mon_lockwait - Sperrwartestatus überwachen (Konfigurationsparameter)
mon_lw_thresh	mon_lw_thresh - Schwellenwert für Sperrenwartestatus überwachen (Konfigurationsparameter)
mon_obj_metrics	mon_obj_metrics - Datenobjektmesswerte überwachen (Konfigurationsparameter)
mon_req_metrics	mon_req_metrics - Anforderungsmesswerte überwachen (Konfigurationsparameter)
mon_uow_data	mon_uow_data - UOW-Ereignisse überwachen (Konfigurationsparameter)
num_iocleaners	num_iocleaners - Anzahl asynchroner Seitenlöschfunktionen (Konfigurationsparameter)
num_ioservers	num_ioservers - Anzahl von E/A-Servern (Konfigurationsparameter)
num_log_span	num_log_span - Anzahl verwendeter Protokolldateien (Konfigurationsparameter)
num_quantiles	num_quantiles - Anzahl der Quantile für Spalten (Konfigurationsparameter)

Tabelle 135. Konfigurierbare memberbezogene Datenbankkonfigurationsparameter (Forts.)

Parameter	Zusätzliche Informationen
numarchretry	numarchretry - Anzahl der Wiederholungen bei Fehler (Konfigurationsparameter)
pckcachesz	pckcachesz - Größe des Paketcache (Konfigurationsparameter)
self_tuning_mem	self_tuning_mem - Speicher mit automatischer Leistungsoptimierung (Konfigurationsparameter)
sheapthres_shr	sheapthres_shr - Schwellenwert für Sortierspeicher für gemeinsame Sortiervorgänge (Konfigurationsparameter)
sortheap	sortheap - Sortierspeichergröße (Konfigurationsparameter)
stat_heap_sz	stat_heap_sz - Größe des Statistikzwischenspeichers (Konfigurationsparameter)
stmt_conc	stmt_conc - Anweisungskonzentrator (Konfigurationsparameter)
stmtheap	stmtheap - Größe des Anweisungszwischenspeichers (Konfigurationsparameter)
tsm_mgmtclass	tsm_mgmtclass - Tivoli Storage Manager-Verwaltungsklasse (Konfigurationsparameter)
tsm_nodename	tsm_nodename - TSM-Knotenname (Konfigurationsparameter)
tsm_owner	tsm_owner - TSM-Eignername (Konfigurationsparameter)
tsm_password	tsm_password - TSM-Kennwort (Konfigurationsparameter)
util_heap_sz	util_heap_sz - Zwischenspeichergröße für Dienstprogramme (Konfigurationsparameter)
wlm_collect_int	wlm_collect_int - Workload-Management-Erfassungsintervall (Konfigurationsparameter)

Erneutes Kompilieren einer Abfrage nach Konfigurationsänderungen

Um die Auswirkungen der Konfigurationsänderungen zu sehen, die die Abfrageoptimierung betreffen, ist es möglicherweise erforderlich, das Abfrageoptimierungsprogramm zu veranlassen, die im Cache gespeicherten Anweisungen erneut zu kompilieren.

Vorgehensweise

Sie können durch Ausführen einer der folgenden Aktionen veranlassen, dass das Abfrageoptimierungsprogramm eine Anweisung erneut kompiliert:

- Machen Sie die im Cache gespeicherten dynamischen Anweisungen für bestimmte Tabellen ungültig, indem Sie folgenden Befehl **RUNSTATS** verwenden:

  ```
  RUNSTATS ON TABLE <tabellenschema>.<tabellenname>
     WITH DISTRIBUTION AND SAMPLED DETAILED INDEXES ALL
  ```

  Anmerkung: Dadurch wird die Tabellenstatistik aktualisiert und nachfolgende Abfragekompilierungen verwenden die neue Statistik sowie die neuen Konfigurationseinstellungen.
- Entfernen Sie alle im Cache gespeicherten dynamischen SQL-Anweisungen, die sich derzeit im Paketcache befinden:

  ```
  FLUSH PACKAGE CACHE DYNAMIC
  ```

Einschränkungen und Verhalten bei der Konfiguration von 'max_coordagents' und 'max_connections'

Der Standardwert für die Parameter **max_coordagents** und **max_connections** bei Version 9.5 ist AUTOMATIC, wobei **max_coordagents** auf 200 und **max_connections** auf -1 gesetzt ist (d. h., dieser Parameter wird auf den Wert von **max_coordagents** gesetzt). Durch diese Einstellungen wird der Konzentrator inaktiviert (OFF).

Bei der Onlinekonfiguration von **max_coordagents** oder **max_connections** sind einige Einschränkungen und Funktionsweisen zu beachten:

- Wenn der Wert von **max_coordagents** erhöht wird, wirkt sich die Einstellung unmittelbar aus und neue Anforderungen können neue koordinierende Agenten erstellen. Wenn der Wert vermindert wird, nimmt die Zahl der koordinierenden Agenten nicht sofort ab. Stattdessen nimmt die Zahl der koordinierenden Agenten nicht weiter zu und vorhandene koordinierende Agenten werden möglicherweise beendet, nachdem sie ihre aktuelle Arbeit erledigt haben, damit die Gesamtzahl der koordinierenden Agenten reduziert wird. Neue Arbeitsanforderungen, für die koordinierende Agenten erforderlich sind, werden erst berücksichtigt, wenn die Gesamtanzahl der koordinierenden Agenten den neuen Wert unterschreitet und ein neuer koordinierender Agenten freigegeben wird.
- Wenn der Wert von **max_connections** erhöht wird, wirkt sich die Einstellung unmittelbar aus und neue Verbindungen, die vorher aufgrund dieses Parameters blockiert wurden, werden zugelassen. Wenn der Wert vermindert wird, beendet der Datenbankmanager vorhandene Verbindungen nicht aktiv. Stattdessen sind neue Verbindungen erst möglich, wenn ausreichend vorhandene Verbindungen beendet wurden, damit der Wert unter das neue Maximum sinkt.
- Wenn **max_connections** auf -1 (Standardwert) gesetzt ist, entspricht die maximal zulässige Anzahl der Verbindungen dem Wert von **max_coordagents**. Wenn **max_coordagents** offline oder online aktualisiert wird, wird auch die maximal zulässige Anzahl der Verbindungen aktualisiert.

Wenn Sie den Wert von **max_coordagents** oder **max_connections** online ändern, können Sie ihn nicht dahingehend ändern, dass der Verbindungskonzentrator aktiviert wird, wenn er inaktiviert ist, bzw. dass der Verbindungskonzentrator inaktiviert wird, wenn er aktiviert ist. Beispiel: Wenn zum Zeitpunkt von **START DBM** der Wert für **max_coordagents** kleiner als der für **max_connections** (Konzentrator ist aktiviert) ist, müssen alle Aktualisierungen, die online für diese beiden Parameter durchgeführt wurden, die Beziehung '**max_coordagents** < **max_connections**' aufrechterhalten. Ähnlich verhält es sich, wenn zum Zeitpunkt von **START DBM** der Wert für **max_coordagents** größer-gleich dem Wert für **max_connections** (Konzentrator ist inaktiviert) ist. In diesem Fall müssen alle online durchgeführten Aktualisierungen diese Beziehung aufrechterhalten.

Wenn Sie diese Art der Aktualisierung online durchführen, lässt der Datenbankmanager die Operation nicht fehlschlagen, sondern verzögert die Aktualisierung. Ähnlich wie in allen Fällen der Aktualisierung der Konfigurationsparameter für den Datenbankmanager, bei denen **IMMEDIATE** angegeben wird, aber nicht möglich ist, wird die Warnung SQL1362W zurückgegeben.

Wenn **max_coordagents** bzw. **max_connections** auf AUTOMATIC gesetzt ist, kann das folgende Verhalten erwartet werden:

- Beide Parameter können mit einem Anfangswert und der Einstellung AUTOMATIC konfiguriert werden. Beispiel: Mit dem folgenden Befehl werden dem Parameter **max_coordagents** der Wert 200 und die Einstellung AUTOMATIC zugeordnet:

  ```
  UPDATE DBM CONFIG USING max_coordagents 200 AUTOMATIC
  ```

  Diesen Parametern ist immer ein Wert zugeordnet. Dabei handelt es sich entweder um den Standardwert oder um einen von Ihnen angegebenen Wert. Ist nur AUTOMATIC bei der Aktualisierung eines der beiden Parameter angegeben, ist also kein Wert angegeben, und war dem Parameter zuvor ein Wert zugeordnet, bleibt dieser Wert bestehen. Dies betrifft nur die Einstellung AUTOMATIC.

Anmerkung: Wenn der Konzentrator aktiviert ist, sind die diesen beiden Konfigurationsparametern zugeordneten Werte auch dann wichtig, wenn die Parameter auf AUTOMATIC gesetzt sind.

- Wenn beide Parameter auf AUTOMATIC gesetzt sind, ermöglicht der Datenbankmanager eine Zunahme der Anzahl der Verbindungen und koordinierenden Agenten je nach Auslastungsbedarf. Allerdings gilt dies nur unter Vorbehalt:
  1. Wenn der Konzentrator inaktiviert ist, behält der Datenbankmanager pro Verbindung nur *einen* einzigen koordinierenden Agenten.
  2. Wenn der Konzentrator aktiviert ist, versucht der Datenbankmanager, das durch die Werte für die Parameter festgelegte Verhältnis von koordinierenden Agenten und Verbindungen zu wahren.

     Anmerkung:

     – Das Verfahren zur Wahrung des Verhältnisses wurde so konzipiert, dass es den Systembetrieb möglichst wenig beeinträchtigt. Es gewährleistet jedoch keine hundertprozentige Beibehaltung des Verhältnisses. In diesem Szenario sind immer neue Verbindungen möglich, auch wenn sie möglicherweise auf einen verfügbaren koordinierenden Agenten warten müssen. Neue koordinierende Agenten werden bei Bedarf zur Aufrechterhaltung des Verhältnisses erstellt. Werden Verbindungen beendet, kann der Datenbankmanager auch koordinierende Agenten zur Aufrechterhaltung des Verhältnisses beenden.
     – Der Datenbankmanager verkleinert das von Ihnen festgelegte Verhältnis nicht. Die von Ihnen festgelegten Anfangswerte für **max_coordagents** und **max_connections** werden als Untergrenze betrachtet.
- Die aktuellen und verzögerten Werte für beide Parameter können z. B. über den CLP oder mithilfe von APIs angezeigt werden. Bei den angezeigten Werten handelt es sich immer um die vom Benutzer festgelegten Werte. Beispiel: Wenn der folgende Befehl abgesetzt und anschließend 30 gleichzeitige Verbindungen gestartet würden, die mit dieser Instanz arbeiteten, würde für **max_connections** und **max_coordagents** weiterhin der Wert 20, AUTOMATIC angezeigt:

  ```
  UPDATE DBM CFG USING max_connections 20 AUTOMATIC,
    max_coordagents 20 AUTOMATIC
  ```

  Wenn Sie die tatsächliche Anzahl der Verbindungen und koordinierenden Agenten ermitteln möchten, die momentan Monitorelemente ausführen, können Sie auch den Diagnosemonitor verwenden.
- Wenn **max_connections** auf AUTOMATIC mit einem größeren Wert als **max_coordagents** gesetzt ist (sodass der Konzentrator aktiviert ist) und **max_coordagents** nicht auf AUTOMATIC gesetzt ist, ermöglicht der Datenbankmanager eine unbegrenzte Anzahl an Verbindungen, die nur eine begrenzte Anzahl an koordinierenden Agenten verwendet.

  Anmerkung: Möglicherweise müssen Verbindungen auf verfügbare koordinierende Agenten warten.

Die Verwendung der Option AUTOMATIC für die Konfigurationsparameter **max_coordagents** und **max_connections** ist nur in den folgenden beiden Szenarios gültig:

1. Für beide Parameter ist die Einstellung AUTOMATIC definiert.
2. Der Konzentrator ist aktiviert, wobei **max_connections** auf AUTOMATIC gesetzt ist, **max_coordagents** dagegen nicht.

Alle anderen Konfigurationen, bei denen AUTOMATIC für diese Parameter verwendet wird, werden blockiert und geben die Nachricht SQL6112N mit einem Ursachencode zurück, in dem die gültigen Einstellungen für AUTOMATIC für diese beiden Parameter erläutert werden.

Konfigurationsparameter des Datenbankmanagers

agent_stack_sz - Größe des Agentenstacks

Dieser Parameter steuert den virtuellen Speicherbereich, den DB2 jedem Agenten zuordnet.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]

Linux (32 Bit)
256 [16 – 1024]

Linux (64 Bit) und UNIX
1024 [256 – 32768]

Windows
16 [8 – 1000]

Anmerkung: Der Standardwert unterliegt nach der ursprünglichen Datenbankerstellung Änderungen durch den DB2 Configuration Advisor.

Maßeinheit
Seiten (4 KB)

Zuordnung
Wenn ein Agent zur Arbeit für eine Anwendung initialisiert wird

Freigabe
Wenn ein Agent die Arbeit für eine Anwendung beendet

Sie können diesen Parameter zur Optimierung der Speicherauslastung des Servers für einen bestimmten Satz von Anwendungen verwenden. Komplexere Abfragen benötigen im Vergleich zu einfachen Abfragen mehr Stackspeicherbereich.

Mit diesem Parameter wird die anfänglich festgeschriebene Stackgröße für jeden Agenten in einer Windows-Umgebung festgelegt. Jeder Agentenstack kann standardmäßig bis auf die Größe des Standardreservestacks anwachsen, d. h. bis auf 256 KB (64 4-KB-Seiten). Dieser Grenzwert ist für die meisten Datenbankoperationen ausreichend. Unter UNIX und Linux wird der Wert des Konfigurationsparameters **agent_stack_sz** auf den nächsthöheren Potenzwert der Basis 2 aufgerundet. Die Standardeinstellung für UNIX sollte für die meisten Auslastungen ausreichend sein. Der Agentenstack hat möglicherweise die Speicherbegrenzung erreicht und es wird 'SQLCODE -973' zurückgegeben.

Der Agentenstack kann vergrößert werden, indem der Parameter **agent_stack_sz** auf einen höheren Wert als die Standardreservestackgröße von 64 Seiten gesetzt wird. Wenn der Wert des Parameters **agent_stack_sz** über der Standardreservestackgröße liegt, wird er vom Windows-Betriebssystem auf das nächste Vielfache von 1 MB gerundet. Wird die Agentenstackgröße beispielsweise auf 128 4-KB-Seiten erhöht, wird für jeden Agenten in Wirklichkeit ein Stack der Größe 1 MB reserviert. Wenn Sie für **agent_stack_sz** einen Wert festlegen, der unter der Standardreservestackgröße liegt, hat dies keine Auswirkungen auf die maximale Begrenzung, da der Agentenstack trotzdem bei Bedarf auf die Größe des Standardreservestacks anwächst. In diesem Fall ist der Wert für **agent_stack_sz** der ursprüngliche festgeschriebene Speicher für den Stack, wenn ein Agent erstellt wird.

Sie können die Standardreservestackgröße ändern, indem Sie mit dem Dienstprogramm **db2hdr** die Headerdaten für die Datei `db2syscs.exe` ändern. Wenn Sie die Standardreservestackgröße ändern, wirkt sich dies auf alle Threads aus, während sich eine Änderung des Werts für **agent_stack_sz** nur auf die Stackgröße für Agenten auswirkt. Das Ändern der Standardstackgröße mithilfe des Dienstprogramms 'db2hdr' bietet den Vorteil einer besseren Granularität, wodurch die Stackgröße auf die minimal erforderliche Stackgröße festgelegt werden kann. Sie müssen jedoch DB2 stoppen und neu starten, damit die an `db2syscs.exe` vorgenommenen Änderungen wirksam werden.

Empfehlung: Wenn Sie beabsichtigen, mit umfangreichen oder komplexen XML-Daten in einer 32-Bit-Umgebung zu arbeiten, sollten Sie den Parameter **agent_stack_sz** auf mindestens `256` 4-KB-Seiten aktualisieren. Bei sehr komplexen XML-Schemata kann es erforderlich sein, den Parameter **agent_stack_sz** wesentlich näher an den Grenzwert zu setzen, um Ausnahmebedingungen wegen Stacküberläufen beim Registrieren von Schemata oder Prüfen von XML-Dokumenten zu vermeiden.

Sie können die Stackgröße eventuell verringern, um anderen Clients mehr Adressraum zur Verfügung zu stellen. Dies ist möglich, wenn Ihre Umgebung folgende Kriterien erfüllt:

- Die Umgebung enthält nur einfache Anwendungen (z. B. einfache Online-Transaktionsprogramme, OLTP), in denen es keine komplexen Abfragen gibt.
- Für die Umgebung sind relativ viele gleichzeitig aktive Clients erforderlich (z. B. über 100).

Unter Windows stehen die Größe des Agentenstacks und die Anzahl gleichzeitig ausgeführter Clients in einem umgekehrten Verhältnis zueinander: Durch eine Vergrößerung des Stacks wird die mögliche Anzahl der Clients verringert, die gleichzeitig ausgeführt werden können. Dies liegt daran, dass der Adressraum unter Windows-Betriebssystemen begrenzt ist.

agentpri - Agentenpriorität

Mit diesem Parameter wird die Priorität gesteuert, die sowohl allen Agenten als auch anderen Prozessen und Threads der Datenbankmanagerinstanz vom Scheduler des Betriebssystems zugewiesen wird. Durch diese Priorität wird festgelegt, wie den Datenbankmanagerprozessen, -agenten und -threads im Vergleich zu den anderen Prozessen und Threads, die auf dem System ausgeführt werden, CPU-Zeit zugewiesen wird.

Wichtig: Der Konfigurationsparameter **agentpri** des Datenbankmanagers gilt seit Version 9.5 als veraltet. Er kann in Datenserver- und -clientversionen vor Version 9.5 weiterhin verwendet werden. Darüber hinaus gilt die Agentenpriorität für die

WLM-Serviceklasse in Version 10.1 als veraltet und wird möglicherweise in einem zukünftigen Release entfernt. Verwenden Sie anstelle der Agentenpriorität die WLM-Dispatcherfunktion. Weitere Informationen hierzu finden Sie im Abschnitt „Die Agentenpriorität der Serviceklassen gilt als veraltet" in *Neuerungen in DB2 Version 10.1*.

Anmerkung: Die nachfolgenden Informationen gelten nur für Datenserver und Clients einer Version vor Version 9.5.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]

AIX -1 (System) [41 - 125]

Andere UNIX-Plattformen
-1 (System) [41 - 128]

Windows
-1 (System) [0 - 6]

Solaris
-1 (System) [0 - 59]

Linux -1 (System) [1 - 99]

HP-UX
-1 (System) [0 - 31]

Wenn der Parameter auf den Wert -1 bzw. System gesetzt ist, wird keine besondere Aktion ausgeführt, und der Datenbankmanager erhält seine CPU-Zeit in der normalen Weise, in der das Betriebssystem allen Prozessen und Threads Prozessorzeit zuweist. Wenn der Parameter auf einen anderen Wert als -1 bzw. System gesetzt wird, erstellt der Datenbankmanager seine Prozesse und Threads mit einer statischen Priorität, die dem Wert des Parameters entspricht. Dadurch können Sie mit diesem Parameter die Priorität steuern, mit der die Prozesse und Threads des Datenbankmanagers (in einer Umgebung mit partitionierten Datenbanken gehören dazu auch koordinierende Agenten und Subagenten, parallele Systemsteuerprogramme und die FCM-Dämonen) auf Ihrem System ausgeführt werden.

Mit diesem Parameter kann der Durchsatz des Datenbankmanager erhöht werden. Die Werte für die Einstellung dieses Parameters sind von dem Betriebssystem abhängig, auf dem der Datenbankmanager ausgeführt wird. Beispielsweise ergeben in einer Linux- oder UNIX-Umgebung niedrige numerische Werte hohe Prioritäten. Wenn der Parameter auf einen Wert zwischen 41 und 125 gesetzt wird, erstellt der Datenbankmanager seine Agenten mit einer statischen UNIX-Priorität, die dem Wert dieses Parameters entspricht. Dies ist in Linux- oder UNIX-Umgebungen von Bedeutung, weil numerisch niedrige Werte hohe Prioritäten für den Datenbankmanager ergeben. Bei anderen Prozessen (einschließlich Anwendungen und Benutzern) können jedoch Verzögerungen auftreten, da sie nicht genügend CPU-Zeit er-

halten. Sie sollten den Wert für diesen Parameter mit den anderen Aktivitäten, die Sie auf der Maschine erwarten, abstimmen.

Einschränkungen:

- Wenn Sie auf Linux- oder UNIX-Plattformen für diesen Parameter einen anderen als den Standardwert verwenden, können Sie den Governor nicht verwenden, um Agentenprioritäten zu ändern.
- Auf dem Solaris-Betriebssystem sollten Sie den Standardwert (-1) nicht ändern. Durch das Ändern des Standardwerts wird die Priorität des DB2-Prozesses auf Echtzeit gesetzt, wodurch alle verfügbaren Ressourcen in dem System monopolisiert werden können.

Empfehlung: Zu Anfang sollte der Standardwert verwendet werden. Dieser Wert stellt einen guten Kompromiss zwischen den Antwortzeiten für andere Benutzer bzw. Anwendungen und dem Durchsatz des Datenbankmanagers dar.

Wenn die Datenbankleistung von Bedeutung ist, können Sie durch Vergleichstests (Benchmark-Tests) die optimale Einstellung für diesen Parameter bestimmen. Eine Erhöhung der Priorität des Datenbankmanagers sollte nur mit großer Vorsicht vorgenommen werden, da die Leistung anderer Benutzerprozesse erheblich beeinträchtigt werden kann, besonders dann, wenn die CPU-Auslastung sehr hoch ist. Durch Erhöhen der Priorität der Datenbankmanagerprozesse und -threads können bedeutende Leistungssteigerungen erzielt werden.

alt_diagpath - Alternativer Verzeichnispfad für Diagnosedaten

Mit diesem Parameter können Sie den vollständig qualifizierten alternativen Pfad für DB2-Diagnosedaten angeben, der verwendet wird, wenn der primäre Pfad für Diagnosedaten (**diagpath**) nicht verfügbar ist.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
Null [beliebiger gültiger Pfadname, , '"*pfadname* $h"', '"*pfadname* $h/*anschließendes_verzeichnis*"', , '"*pfadname* $n"',[2] '"*pfadname* $n/*anschließendes_verzeichnis*"', '"*pfadname* $m"', '"*pfadname* $m/*anschließendes_verzeichnis*"', '"*pfadname* hn"',[3] '"*pfadname* hn/*anschließendes_verzeichnis*"', '"*pfadname* hm"' oder '"*pfadname* hm/*anschließendes_verzeichnis*"']

Symbole

2. $n ist veraltet und wird in einem zukünftigen Release möglicherweise entfernt.

3. hn ist veraltet und wird in einem zukünftigen Release möglicherweise entfernt.

pfadname
Ein Verzeichnispfad, der verwendet wird, wenn der primäre Verzeichnispfad für Diagnosedaten nicht verfügbar ist.

$h Wird in HOST_*hostname* aufgelöst.

Anmerkung: Ab Version 10 bezieht sich $h in DB2 pureScale-Umgebungen auf den Benutzerhost des Members.

$n Wird in NODE*nummer* aufgelöst.

$m Wird in DIAG_*nummer* aufgelöst. Dabei ist zu beachten, dass die Verwendung von DIAG_*nummer* nicht davon abhängig ist, ob dieser Wert auf eine Datenbankpartition verweist, eine CF oder auf ein Member.

/anschließendes_verzeichnis
Ein einzelnes Verzeichnis oder ein Verzeichnis mit Unterverzeichnis, das sich an die Angabe $h oder $n anschließt.

Die folgenden Werte sind verfügbar:

- '"*pfadname* $h"'
- '"*pfadname* $h/*anschließendes_verzeichnis*"'
- '"*pfadname* $n"'
- '"*pfadname* $n/*anschließendes_verzeichnis*"'
- '"*pfadname* $m"'
- '"*pfadname* $m/*anschließendes_verzeichnis*"'
- '"*pfadname* hn"'
- '"*pfadname* hn/*anschließendes_verzeichnis*"'
- '"*pfadname* hm"'
- '"*pfadname* hm/*anschließendes_verzeichnis*"'

Das alternative Verzeichnis für Diagnosedaten kann dieselben Diagnosedaten wie das primäre Verzeichnis für Diagnosedaten enthalten, das mit dem Parameter **diagpath** festgelegt wird. Wenn der Parameter **alt_diagpath** definiert wird und das primäre Verzeichnis für Diagnosedaten nicht verfügbar ist, läuft die Protokollierung von Diagnosedaten im angegebenen alternativen Verzeichnispfad für Diagnosedaten weiter und wird dann an der ursprünglichen Position fortgesetzt, wenn der primäre Diagnosepfad wieder verfügbar ist. Wenn dieser Parameter null ist und das primäre Verzeichnis für Diagnosedaten, das über den Parameter **diagpath** angegeben wird, nicht verfügbar ist, werden keine weiteren Diagnosedaten aufgezeichnet, bis der primäre Diagnosepfad wieder verfügbar ist. Eine bessere Ausfallsicherheit erreichen Sie dadurch, dass Sie das alternative Verzeichnis für Diagnosedaten so definieren, dass es auf ein anderes Dateisystem als das primäre Verzeichnis für Diagnosedaten verweist.

Ab Version 10.1 schreibt der alternative Verzeichnispfad für Diagnosedaten für jedes Member und jede CF standardmäßig in die private Datei 'db2diag.log'. Zum Zurücksetzen auf das Verhalten der früheren Releases, in denen die Diagnosedaten in dasselbe Verzeichnis geschrieben wurden, geben Sie **alt_diagpath** mit einem *pfadnamen* und ohne Token ($h, $n oder $m) an.

Anmerkung:

- Damit die Betriebssystemshell das Zeichen $ auf Linux- und UNIX-Systemen nicht als Steuerzeichen interpretiert, müssen außerhalb der doppelten Anführungszeichen zusätzlich einfache Anführungszeichen (Hochkommas) eingegeben werden, wie aus dem Syntaxbeispiel hervorgeht.

- Im CLP-Dialogmodus - oder wenn der Befehl aus einer Eingabedatei gelesen und ausgeführt wird - ist das doppelte Anführungszeichen nicht erforderlich.
- Bei `$h`, `$m` und `$n` wird nicht zwischen Groß- und Kleinschreibung unterschieden.
- Das dynamische Verhalten für **alt_diagpath** erstreckt sich nicht auf alle Prozesse.
- Der DB2-Serverprozess **db2sysc** kann dynamische Änderungen erkennen, z. B. wenn der Befehl **UPDATE DATABASE MANAGER CONFIGURATION** für eine Instanzzuordnung ausgegeben wird.
- Wenn DB2-Client- und -Anwendungsprozesse gestartet werden, wird dafür die Einstellung des Konfigurationsparameters **alt_diagpath** verwendet und es werden keine dynamischen Änderungen erkannt.
- Wenn auf UNIX-Systemen weder **diagpath** noch **alt_diagpath** verfügbar ist, wird die db2-Diagnosenachricht in der syslog-Datei gespeichert.
- Es gibt kein Standardverzeichnis für den Konfigurationsparameter **alt_diagpath**.
- Die Konfigurationsparameter **alt_diagpath** und **diagpath** schließen einander aus. Sie können nicht auf denselben Verzeichnispfad gesetzt werden.
- Wenn **alt_diagpath** (oder **diagpath**) nicht verfügbar ist, bedeutet dies, dass das Speichern von Diagnosedaten aufgrund eines Fehlers fehlgeschlagen ist, wie: das Verzeichnis wurde gelöscht, es gab einen Plattenfehler, die Platte ging verloren, es gab Netzprobleme, es gab einen Dateiberechtigungsfehler oder die Platte ist voll.

alternate_auth_enc - Alternativer Verschlüsselungsalgorithmus für ankommende Verbindungen auf dem Server (Konfigurationsparameter)

Mit diesem Konfigurationsparameter wird der alternative Verschlüsselungsalgorithmus angegeben, der zur Verschlüsselung der Benutzer-IDs und der Kennwörter verwendet wird, die an einen DB2-Datenbankserver zur Authentifizierung übergeben werden. Insbesondere beeinflusst dieser Parameter den Verschlüsselungsalgorithmus, wenn die zwischen dem DB2-Client und dem DB2-Datenbankserver vereinbarte Authentifizierungsmethode SERVER_ENCRYPT ist.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
`NOT_SPECIFIED [AES_CMP; AES_ONLY]`

Die Benutzer-ID und das Kennwort, die zur Authentifizierung an den DB2-Datenbankserver übergeben werden, werden verschlüsselt, wenn die zwischen dem DB2-Client und dem DB2-Server vereinbarte Authentifizierungsmethode SERVER_ENCRYPT ist. Die vereinbarte Authentifizierungsmethode hängt von der Einstellung für den Authentifizierungstyp auf dem Server und dem vom Client angeforderten Authentifizierungstyp ab. Die Auswahl des zur Verschlüsselung der Benutzer-ID und des Kennworts verwendeten Verschlüsselungsalgorithmus hängt von der Einstel-

lung des Konfigurationsparameters **alternate_auth_enc** des Datenbankmanagers ab. Je nach dem Wert dieser Einstellung kann DES- oder AES-Verschlüsselung verwendet werden.

Wenn der Standardwert (NOT_SPECIFIED) verwendet wird, akzeptiert der Server den Verschlüsselungsalgorithmus, den der Client vorschlägt.

Wenn der Parameter **alternate_auth_enc** auf den Wert AES_ONLY gesetzt wird, akzeptiert der Datenbankserver nur Verbindungen, die mit AES-Verschlüsselung arbeiten. Wenn der Client die AES-Verschlüsselung nicht unterstützt, wird die Verbindung zurückgewiesen.

Wenn der Parameter **alternate_auth_enc** auf den Wert AES_CMP gesetzt wird, akzeptiert der Datenbankserver Benutzer-IDs und Kennwörter, die entweder mit AES oder DES verschlüsselt wurden, vereinbart jedoch AES, wenn der Client die AES-Verschlüsselung unterstützt.

aslheapsz - Zwischenspeichergröße für Anwendungsunterstützungsebene

Der Zwischenspeicher für die Anwendungsunterstützungsebene ist ein Kommunikationspuffer zwischen der lokalen Anwendung und dem zugeordneten Agenten. Dieser Puffer wird als gemeinsam benutzter Speicher von jedem Datenbankmanageragenten, der gestartet wird, zugeordnet.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
15 [1 - 524 288]

Anmerkung: Der Standardwert unterliegt nach der ursprünglichen Datenbankerstellung Änderungen durch den DB2 Configuration Advisor.

Maßeinheit
Seiten (4 KB)

Zuordnung
Wenn der Agentenprozess des Datenbankmanagers für die lokale Anwendung gestartet wird

Freigabe
Wenn der Agentenprozess des Datenbankmanagers beendet wird

Wenn die Anforderung an den Datenbankmanager oder die zugehörige Antwort nicht in den Puffer passt, wird sie in zwei oder mehr Sende-/Empfangspufferpaare aufgeteilt. Die Größe dieses Puffers sollte so festgelegt werden, dass die Mehrzahl der Anforderungen mit einem einzigen Sende-/Empfangspufferpaar verarbeitet werden kann. Die Größe der Anforderung hängt von der Speichermenge ab, die zur Speicherung folgender Daten erforderlich ist:

- Der Eingabe-SQL-Deskriptorbereich

- Alle zugeordneten Daten in den SQLVARs
- Der Ausgabe-SQL-Deskriptorbereich
- Andere Felder, die im Allgemeinen 250 Byte nicht überschreiten

Außer zur Steuerung dieses Kommunikationspuffers dient dieser Parameter auch den beiden folgenden Zwecken:

- Dieser Parameter wird verwendet, um die E/A-Blockgröße festzulegen, wenn ein Blockcursor geöffnet wird. Dieser Speicher für Blockcursor wird aus dem privaten Adressraum der Anwendung zugeordnet. Sie sollten daher die optimale Größe des privaten Speichers ermitteln, der jedem Anwendungsprogramm zugeordnet werden soll. Wenn der Data Server Runtime Client keinen Bereich für einen Blockcursor aus dem privaten Speicher der Anwendung zuordnen kann, wird ein Cursor ohne Blockung geöffnet.
- Dieser Parameter wird verwendet, um die Kommunikationsgröße zwischen Agenten und **db2fmp**-Prozessen festzulegen. (Ein **db2fmp**-Prozess kann eine benutzerdefinierte Funktion oder eine abgeschirmte gespeicherte Prozedur sein.) Die Anzahl Byte wird jedem **db2fmp**-Prozess bzw. jedem im System aktiven Thread aus dem gemeinsamen Speicher zugeordnet.

Die Daten, die von der lokalen Anwendung gesendet werden, werden vom Datenbankmanager in einem Bereich zusammenhängenden Speichers empfangen, der aus dem Abfragezwischenspeicher zugeordnet wird. Mit dem Parameter **aslheapsz** wird die Anfangsgröße des Abfragezwischenspeichers (für lokale und ferne Clients) festgelegt. Die maximale Größe des Abfragezwischenspeichers wird durch den Parameter **query_heap_sz** definiert.

Empfehlung: Wenn die Anforderungen Ihrer Anwendung im Allgemeinen klein sind und die Anwendung auf einem System mit eingeschränkter Speicherkapazität ausgeführt wird, ist es möglicherweise sinnvoll, den Wert dieses Parameters zu verringern. Wenn Ihre Abfragen im Allgemeinen sehr groß sind, mehr als eine Sende- und Empfangsanforderung erfordern und die Speicherkapazität Ihres Systems nicht eingeschränkt ist, kann es sinnvoll sein, den Wert dieses Parameters zu erhöhen.

Berechnen Sie anhand der folgenden Formel die Mindestanzahl der Seiten für **aslheapsz**:

```
aslheapsz >= ( sizeof(Eingabe-SQL-Deskriptorbereich)
             + sizeof(jeder Eingabe-SQLVAR)
             + sizeof(Ausgabe-SQL-Deskriptorbereich)
             + 250 ) / 4096
```

Dabei ist sizeof(x) die Größe von x in Byte zur Berechnung der Seitenanzahl eines bestimmten Eingabe- oder Ausgabewerts.

Beachten Sie außerdem, welche Auswirkung dieser Parameter auf die Anzahl und die mögliche Größe von Blockcursorn hat. Große Zeilenblöcke können zu einer höheren Leistung führen, wenn die Anzahl oder die Größe der Zeilen, die übertragen werden, groß ist (z. B., wenn die Datenmenge größer als 4.096 Byte ist). Dies hat jedoch den Nachteil, dass größere Datensatzblöcke die Größe des für jede Verbindung benötigten Arbeitsspeichers erhöhen.

Größere Satzblöcke können außerdem mehr Blockabrufanforderungen verursachen, als für die Anwendung erforderlich wären. Sie können die Anzahl der Abrufanforderungen mit der Klausel OPTIMIZE FOR in der Anweisung SELECT in Ihrer Anwendung steuern.

audit_buf_sz - Prüfpuffergröße

Mit diesem Parameter wird die Größe des Puffers für die Prüfung der Datenbank angegeben.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
`0 [0 - 65 000 ]`

Maßeinheit
Seiten (4 KB)

Zuordnung
Beim Starten von DB2

Freigabe
Beim Stoppen von DB2

Der Standardwert für diesen Parameter ist Null (`0`). Wenn der Wert Null (`0`) ist, wird der Prüfpuffer nicht verwendet. Wenn der Wert größer als Null (0) ist, wird dem Prüfpuffer Speicherbereich zugeordnet, in den die von der Prüffunktion generierten Prüfsätze gestellt werden. Der Wert multipliziert mit 4-KB-Seiten ergibt die für den Prüfpuffer zugeordnete Speichermenge. In regelmäßigen Zeitintervallen oder bei vollem Prüfpuffer führt der Prüfprotokolldämonprozess **db2auditd** eine Flushoperation für den Prüfpuffer auf Platte aus. Der Prüfpuffer kann nicht dynamisch zugeordnet werden; DB2 muss gestoppt und anschließend erneut gestartet werden, bevor der neue Wert für diesen Parameter in Kraft tritt.

Wenn Sie den Standardwert für diesen Parameter in einen Wert ändern, der größer als Null (0) ist, schreibt die Prüffunktion Datensätze asynchron im Vergleich zur Ausführung der Anweisungen, die die Prüfsätze generieren, auf Platte. Durch das Erhöhen des Standardparameterwerts (`0`) wird die Leistung von DB2 verbessert. Der Wert Null (`0`) bedeutet, dass die Prüffunktion Datensätze synchron zur (d. h. zur gleichen Zeit wie die) Ausführung der Anweisungen, die die Prüfsätze generieren, auf Platte schreibt. Die synchrone Verarbeitung während der Prüfung verringert die Leistung von unter DB2 ausgeführten Anwendungen.

authentication - Authentifizierungstyp

Mit diesem Parameter wird angegeben und festgelegt, wie und wo die Authentifizierung eines Benutzers stattfindet.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
SERVER [CLIENT; SERVER; SERVER_ENCRYPT; DATA_ENCRYPT; DATA_ENCRYPT_CMP; KERBEROS; KRB_SERVER_ENCRYPT; GSSPLUGIN; GSS_SERVER_ENCRYPT]

Wird für **authentication** der Wert SERVER angegeben, werden die Benutzer-ID und das Kennwort vom Client an den Server gesendet, sodass die Authentifizierung auf dem Server ausgeführt werden kann. Der Wert SERVER_ENCRYPT unterscheidet sich vom Wert SERVER nur darin, dass alle über das Netz gesendeten Benutzer-IDs und Kennwörter verschlüsselt werden.

Der Wert DATA_ENCRYPT bedeutet, dass der Server verschlüsselte SERVER-Authentifizierungsschemata und die Verschlüsselung von Benutzerdaten akzeptiert. Die Authentifizierung funktioniert in exakt gleicher Weise wie bei SERVER_ENCRYPT.

Die folgenden Benutzerdaten werden bei Verwendung dieses Authentifizierungstyps verschlüsselt:

- SQL- und XQuery-Anweisungen
- Daten von SQL-Programmvariablen.
- Ausgabedaten aus der Serververarbeitung einer SQL- oder XQuery-Anweisung, einschließlich einer Beschreibung der Daten.
- Einige oder alle Antwortgruppendaten, die aus einer Abfrage resultieren.
- LOB-Streaming (LOB, Large Objects).
- SQLDA-Deskriptoren.

Der Wert DATA_ENCRYPT_CMP bedeutet, dass der Server verschlüsselte SERVER-Authentifizierungsschemata und die Verschlüsselung von Benutzerdaten akzeptiert. Darüber hinaus bietet dieser Authentifizierungstyp Kompatibilität mit Produkten früherer Versionen, die den Authentifizierungstyp DATA_ENCRYPT nicht unterstützen. Diese Produkte erhalten die Möglichkeit, die Verbindung mit dem Authentifizierungstyp SERVER_ENCRYPT und ohne Verschlüsselung von Benutzerdaten herzustellen. Produkte, die den neuen Authentifizierungstyp unterstützen, müssen ihn verwenden. Dieser Authentifizierungstyp ist nur in der Konfigurationsdatei des Datenbankmanagers des Servers, jedoch nicht im Befehl **CATALOG DATABASE** gültig.

Anmerkung: Für eine Konfiguration der Einhaltung von Standards (definiert im Abschnitt zur „Einhaltung von Standards") ist SERVER der einzige unterstützte Wert.

Der Wert CLIENT gibt an, dass alle Authentifizierungen auf dem Client stattfinden. Auf dem Server muss keine Authentifizierung mehr ausgeführt werden.

Der Wert KERBEROS bedeutet, dass die Authentifizierung auf einem Kerberos-Server mithilfe des Kerberos-Sicherheitsprotokolls für Authentifizierung ausgeführt wird. Bei Verwendung des Authentifizierungstyps KRB_SERVER_ENCRYPT auf dem Server und Unterstützung des Kerberos-Sicherheitssystems durch die Clients ist der tatsächliche Systemauthentifizierungstyp KERBEROS. Unterstützen die Clients das Kerberos-Sicherheitssystem nicht, ist der Systemauthentifizierungstyp praktisch äquivalent zu SERVER_ENCRYPT.

Der Wert GSSPLUGIN bedeutet, dass die Authentifizierung durch einen externen GSSAPI-basierten Sicherheitsmechanismus ausgeführt wird. Bei Verwendung des

Authentifizierungstyps GSS_SERVER_ENCRYPT auf dem Server und Unterstützung des GSSPLUGIN-Sicherheitsmechanismus durch die Clients ist der tatsächliche Systemauthentifizierungstyp GSSPLUGIN (d. h., wenn die Clients eines der Plug-ins des Servers unterstützen). Unterstützen die Clients den GSSPLUGIN-Sicherheitsmechanismus nicht, ist der Systemauthentifizierungstyp praktisch äquivalent zu SERVER_ENCRYPT.

Empfehlung: In der Regel ist der Standardwert (SERVER) für lokale Clients geeignet. Wenn ferne Clients eine Verbindung zum Datenbankserver herstellen, ist SERVER_ENCRYPT der empfohlene Wert zum Schutz der Benutzer-ID und des Kennworts.

CF_diaglevel - Aufzeichnungsebene bei Fehlerdiagnose (Konfigurationsparameter für CF)

Dieser Parameter gibt den Typ der Diagnosefehler an, die in den Dateien cfdiag*.log aufgezeichnet werden.

Konfigurationstyp
Datenbankmanager

Gilt für

- DB2 pureScale

Parametertyp
Offline konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
2 [1 — 4]

Gültige Werte für diesen Parameter sind:

- 0 – Keine Aufzeichnung von Diagnosedaten
- 1 – Nur schwerwiegende Fehler
- 2 – Alle Fehler
- 3 – Alle Fehler und Warnungen
- 4 – Alle Fehler, Warnungen und Informationsnachrichten

Der Konfigurationsparameter **cf_diagpath** wird zur Angabe des Verzeichnisses verwendet, in dem sich die Diagnosenachrichtendatei befindet, die in Abhängigkeit vom Wert des Parameters **cf_diaglevel** generiert wird.

Die Position der CF-Protokolldatei wird in der folgenden Reihenfolge festgelegt:

- **cf_diagpath** (DBM-Konfigurationsparameter)
- **diagpath** (DBM-Konfigurationsparameter)
- **DB2PATH**/db2dump

CF_diagpath - Verzeichnispfad für Diagnosedaten für CF (Konfigurationsparameter)

Mit diesem Parameter können Sie einen vollständig qualifizierten Pfad für die Diagnosedatendatei für CF angeben.

Konfigurationstyp
Datenbankmanager

Gilt für

- DB2 pureScale

Parametertyp
Offline konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
INSTHOME/sqllib/db2dump/$m [beliebiger gültiger Pfadname]

Ab Version 10.1 schreibt der Verzeichnispfad für CF-Diagnosedaten für jede CF standardmäßig in die private Datei 'db2diag.log'. Zum Zurücksetzen auf das Verhalten der früheren Releases, in denen die Diagnosedaten für die CF in dasselbe Verzeichnis geschrieben wurden, geben Sie **cf_diagpath** mit einem *pfadnamen* und ohne Token an.

Jeder Name einer CF-Protokolldatei hat folgendes Format:

```
cfdiag-JJJJMMTThhmmssuuuuuu.<cf#>.log
```

Außerdem gibt es einen einzigen Namen eines statischen CF-Diagnoseprotokolls, der stets auf die aktuellste CF-Diagnoseprotokolldatei verweist und folgendes Format hat:

```
cfdiag.<cf#>.log
```

Wenn Sie beispielsweise alle CF-Protokolldateien auflisten, würde das Ergebnis ungefähr so aussehen:

```
$ ls -la cfdiag*
lrwxrwxrwx 1 db2inst1 pdxdb2   35 2011-02-09 15:07 cfdiag.128.log ->
  cfdiag-20110209150733000049.128.log
lrwxrwxrwx 1 db2inst1 pdxdb2   35 2011-02-09 15:10 cfdiag.129.log ->
  cfdiag-20110209151021000040.129.log
-rw-r-xr-- 1 db2inst1 pdxdb2 1271 2011-02-09 15:07 cfdiag-20110209150733000049.128.log
-rw-r-xr-- 1 db2inst1 pdxdb2 1271 2011-02-09 15:07 cfdiag-20110209150740000082.129.log
-rw-r-xr-- 1 db2inst1 pdxdb2 1274 2011-02-09 15:10 cfdiag-20110209151021000040.129.log
```

In diesem Beispiel sind `cfdiag.128.log` und `cfdiag.129.log` symbolische Verbindungen zur neuesten Version der ansonsten mit Zeitmarken versehenen Protokolldateien.

Die CF-Protokolldatei ähnelt der Datei `db2diag.log`.

CF_mem_sz - CF-Speicher (Konfigurationsparameter)

Dieser Parameter steuert den Gesamtspeicher für die Cluster-Caching-Funktion (auch als CF bezeichnet).

Konfigurationstyp
Datenbankmanager

Gilt für
Gilt nur für eine DB2 pureScale-Instanz.

- Datenbankserver mit lokalen und fernen Clients

Parametertyp
Offline konfigurierbar

Standardwert [Bereich]
AUTOMATIC [32768 - 4 294 967 295]

Maßeinheit
Seiten (4 KB)

Zuordnung
Wenn die CF gestartet wird

Freigabe
Wenn die CF gestoppt wird

Bei Verwendung der Standardeinstellung AUTOMATIC wird die Kapazität des Speichers der Cluster-Caching-Funktion durch Abfragen des verfügbaren Gesamtspeichers auf dem CF-Server ermittelt. Anschließend wird der Parameter auf den kleineren von zwei Werten gesetzt: entweder auf einen geeigneten Prozentsatz vom Gesamtspeicher des CF-Servers oder auf die Größe des freien Speichers auf der Maschine. Ein geeigneter Prozentsatz beträgt in der Regel 70 % bis 90 % des verfügbaren Gesamtspeichers auf der CF. Die folgenden Faktoren fließen ebenfalls in die Berechnung unter der Einstellung AUTOMATIC mit ein:

- Ob die CF und irgendwelche DB2-Member koexistieren.
- Ob mehrere Instanzen auf demselben Host vorhanden sind.

cf_num_conns - Anzahl der CF-Verbindungen pro Member in CF (Konfigurationsparameter)

Dieser Parameter steuert die Anfangsgröße des Verbindungspools der Cluster-Caching-Funktion (CF).

Konfigurationstyp
Datenbankmanager

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
AUTOMATIC [4 - 256]

Zuordnung
Beim Starten von DB2

Freigabe
Beim Stoppen von DB2

Wenn der Parameter **cf_num_conns** auf AUTOMATIC (Standardwert) gesetzt ist, erstellt der DB2-Datenbankmanager beim Starten eine Anfangsanzahl von CF-Verbindungen für jedes Member in jeder CF. Diese Anfangsanzahl basiert auf der Anzahl der Worker-Threads, (siehe „CF_num_workers - Anzahl Worker-Threads (Konfigurationsparameter)" auf Seite 764), der Verbindungsanzahl pro Worker-Thread und der Anzahl der Member im Cluster. Der tatsächliche Maximalwert für den Parameter **cf_num_conns** wird beim Memberstart auf der Grundlage dieser Parameter automatisch von DB2 berechnet.

Wenn der Parameter **cf_num_conns** auf AUTOMATIC gesetzt ist und danach durch eine Instanzverbindung auf einen höheren als den aktuellen Wert heraufgesetzt wird, werden neue CF-Verbindungen erstellt. Wenn Sie jedoch für den Parameter **cf_num_conns** einen niedrigeren als den aktuellen Wert festlegen, werden keine CF-Verbindungen beendet.

Wenn der Parameter **cf_num_conns** auf AUTOMATIC gesetzt ist, werden alle aufgrund eines Portausfalls (oder eines anderen Netzproblems) gelöschten Verbindun-

gen auf die verbleibenden Ports verteilt und wiederhergestellt. Sobald ein ausgefallener Port wieder online ist, werden die Verbindungen nach Bedarf neu verteilt.

Wenn der Parameter **cf_num_conns** mit einem festen numerischen Wert definiert wird, erstellt der DB2-Datenbankmanager beim Starten genau diese Anzahl von CF-Verbindungen für jedes Member in jedem CF. Der DB2-Datenbankmanager erhöht bzw. verringert die Verbindungsanzahl nicht automatisch.

Wenn der Parameter **cf_num_conns** auf einen festen Wert gesetzt ist und danach durch eine zusätzliche Instanzverbindung erhöht oder verringert wird, wird die Anzahl der CF-Verbindungen sofort an den neuen Wert angepasst (d. h. erhöht oder verringert).

Wenn der Parameter **cf_num_conns** auf einen festen Wert gesetzt ist, wird bei einem Portausfall nicht die Anzahl der Verbindungen an den verbleibenden Ports erhöht. Um die Anzahl der auf jedem Port aktiven Verbindungen zu ermitteln, wird der Parameterwert für **cf_num_conns** durch die Anzahl der Ports dividiert. Diese Anzahl bleibt unverändert, auch wenn manche Ports offline sind.

CF_num_workers - Anzahl Worker-Threads (Konfigurationsparameter)

Der Parameter **cf_num_workers** gibt die Gesamtzahl von Worker-Threads für die Cluster-Caching-Funktion (CF) an. Worker-Threads werden auf die Schnittstellen für Clusterverbindungen verteilt, um die Anzahl der Worker-Threads auszugleichen, die für die einzelnen Schnittstellen Anforderungen ausführen.

Konfigurationstyp
Datenbankmanager

Gilt für
Gilt nur für eine DB2 pureScale-Instanz.

Parametertyp
Offline konfigurierbar

Standardwert [Bereich]
AUTOMATIC [1 - 31]

Bei einer Einstellung auf den Standardwert (`AUTOMATIC`) wird der Parameterwert so konfiguriert, dass die Anzahl der für die CF verfügbaren Prozessoren um eins unterschritten wird. Die verfügbaren CPUs werden gleichmäßig auf die Instanzen verteilt, bevor von dem sich für die einzelnen Instanzen ergebenden Wert ein Prozessor abgezogen wird. Wenn auf dem Host der CF mehrere Member vorhanden sind, wird die Anzahl der Worker-Threads für die CFen weiter durch die Gesamtzahl der CFen und Member auf dem Host dividiert.

Wenn auf der Servermaschine der CF nur ein Prozessor vorhanden ist, wird dieser Wert auf 1 gesetzt.

Die Gesamtzahl der CF-Worker-Threads wird in der Datei `cfdiag.log` angezeigt.

Dividieren Sie zum Berechnen der Anzahl von Worker-Threads, die jeder Schnittstelle für Clusterverbindungen zugeordnet ist, die Gesamtzahl der CF-Worker-Threads durch die Anzahl der Schnittstellen für Clusterverbindungen, die für die CF definiert sind.

```
Anzahl der Worker-Threads, die jeder Schnittstelle zugeordnet sind =
(cf_num_workers) / (Anzahl der Schnittstellen)
```

Wenn für den Parameter **cf_num_workers** der Wert AUTOMATIC festgelegt ist, konfiguriert der Datenbankmanager die Anzahl der Worker-Threads so, dass bei einer Teilung durch die Anzahl der Schnittstellen für Clusterverbindungen, die für die CF definiert sind, gleich hohe Werte entstehen.

Auf einer CF-Servermaschine, auf der 7 verfügbare Prozessoren und 2 Schnittstellen für Clusterverbindungen vorhanden sind, lautet der Wert für die Einstellung AUTOMATIC wie folgt:

```
Gesamtzahl der Worker-Threads = (7 Prozessoren) - (1 Prozessor, um zu verhindern,
dass alle Prozessoren verwendet werden) = 6
Daraus folgt: Anzahl der Worker-Threads, die mit den einzelnen Schnittstellen
für Clusterverbindungen verbunden sind = 6 / 2 = 3
```

Wenn Sie den Parameter **cf_num_workers** manuell festlegen, müssen Sie den Wert so festlegen, dass er gleich oder größer als die Anzahl der Schnittstellen für Clusterverbindungen ist, sodass mindestens ein Worker-Thread pro Schnittstelle vorhanden ist. Wenn zum Abdecken aller Schnittstellen nur eine nicht ausreichende Anzahl von Worker-Threads vorhanden ist, wird für die CF, deren Start fehlschlägt, ein Alert protokolliert. Der Parameterwert muss geändert werden, damit das Problem behoben wird.

Einschränkungen:

Wenn Sie mit InfiniBand oder uDAPL arbeiten, sollte dieser Wert nicht größer als die Anzahl der Prozessoren auf der CF-Servermaschine sein. Jeder Worker-Thread wird auf einem Prozessor ausgeführt und wartet auf uDAPL-Kommunikation. Die Leistung ist beeinträchtigt, wenn die Anzahl der Worker-Threads die Anzahl der Prozessoren übersteigt.

catalog_noauth - Katalogisieren ohne Berechtigung zulässig

Dieser Parameter gibt an, ob Benutzer Datenbanken und Knoten oder DCS- und ODBC-Verzeichnisse ohne SYSADM-Berechtigung katalogisieren und aus dem Katalog entfernen können.

Konfigurationstyp

Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp

Online konfigurierbar

Weitergabeklasse

Sofort

Standardwert [Bereich]

Datenbankserver mit lokalen und fernen Clients

NO [NO (0) — YES (1)]

Client; Datenbankserver mit lokalen Clients

YES [NO (0) — YES (1)]

Der Standardwert (0) für diesen Parameter gibt an, dass SYSADM-Berechtigung erforderlich ist. Wenn dieser Parameter auf 1 gesetzt ist, ist keine SYSADM-Berechtigung erforderlich.

clnt_krb_plugin - Client-Kerberos-Plug-in

Mit diesem Parameter wird der Name der Standard-Plug-in-Bibliothek für Kerberos angegeben, die für die clientseitige Authentifizierung sowie für die lokale Berechtigung zu verwenden ist.

Konfigurationstyp
Datenbankmanager

Gilt für

- Client
- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit ausschließlich lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
Null unter Linux- und UNIX-Betriebssystemen und `IBMkrb5` unter Windows-Betriebssystemen [beliebige gültige Zeichenfolge].

Das Plug-in wird verwendet, wenn der Client mithilfe der Kerberos-Authentifizierung authentifiziert wird oder wenn die lokale Authentifizierung ausgeführt wird und als Authentifizierungstyp in der Datenbankkonfiguration `KERBEROS` festgelegt ist.

clnt_pw_plugin - Client-Plug-in für Benutzer-ID und Kennwort

Mit diesem Parameter wird der Name der Benutzer-ID/Kennwort-Plug-in-Bibliothek angegeben, die für die clientseitige Authentifizierung sowie für die lokale Berechtigung zu verwenden ist.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
NULL [beliebige gültige Zeichenfolge]

Standardmäßig ist dieser Wert null und die von DB2 bereitgestellte Benutzer-ID/Kennwort-Plug-in-Bibliothek wird verwendet. Das Plug-in wird verwendet, wenn der Client mit dem Authentifizierungstyp CLIENT authentifiziert wird oder wenn die lokale Authentifizierung durchgeführt und der Parameter **authentication** in der Datenbankmanagerkonfiguration (DBM CFG) `CLIENT`, `SERVER`, `SERVER_ENCRYPT`, `DATA_ENCRYPT` oder `DATA_ENCRYPT_CMP` lautet. Für Nichtrootinstallationen gilt, dass bei Verwendung der DB2-Bibliothek für Plug-ins für Benutzer-ID und Kennwort der Befehl **db2rfe** ausgeführt werden muss, bevor das DB2-Datenbankprodukt verwendet wird.

cluster_mgr - Name des Cluster-Managers

Mithilfe dieses Parameters kann der Datenbankmanager inkrementelle Änderungen an der Clusterkonfiguration an den angegebenen Cluster-Manager übertragen.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Informativ

Standardwert

- In einer DB2 pureScale-Umgebung: `TSA`; sonst kein Standardwert

Gültige Werte

- `TSA`

Dieser Parameter wird beim Installieren von DB2 pureScale Feature oder bei Verwendung von DB2 High Availability Instance Configuration Utility (**db2haicu**) automatisch festgelegt, um den Cluster mit hoher Verfügbarkeit zu konfigurieren.

comm_bandwidth - Kommunikationsbandbreite

Dieser Parameter unterstützt das Abfrageoptimierungsprogramm bei der Ermittlung von Zugriffspfaden, indem er die Bandbreite zwischen Datenbankpartitionsservern angibt.

Konfigurationstyp
Datenbankmanager

Gilt für
Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Anweisungsgrenzwert

Standardwert [Bereich]
`-1` [`-1`, `0.1 - 100000`]

Der Wert `-1` bewirkt, dass der Parameter auf den Standardwert zurückgesetzt wird. Der Standardwert wird ausgehend von der Geschwindigkeit der zugrunde liegenden Kommunikation berechnet. Ein Wert von `100` kann für Systeme erwartet werden, die Gigabit Ethernet verwenden.

Maßeinheit
Megabyte pro Sekunde

Der Wert, der (in MB pro Sekunde) für die Übertragungsbandbreite berechnet wird, wird vom Abfrageoptimierungsprogramm zur Abschätzung des Aufwands für bestimmte Operationen zwischen den Datenbankpartitionsservern eines partitionierten Datenbanksystems herangezogen. Das Optimierungsprogramm modelliert nicht die Kosten der Datenfernverarbeitung zwischen einem Client und einem Server. Dieser Parameter sollte daher nur die nominale Bandbreite zwischen den Datenbankpartitionsservern darstellen.

Sie können diesen Wert explizit festlegen, um ein Modell einer Produktionsumgebung auf Ihrem Testsystem zu erstellen oder die Auswirkungen einer Hardwareaufrüstung zu bewerten.

Empfehlung: Sie sollten diesen Parameter nur anpassen, wenn Sie ein Modell einer anderen Umgebung erstellen wollen.

Die Übertragungsbandbreite wird vom Optimierungsprogramm bei der Bestimmung der Zugriffspfade verwendet. Wenn Sie diesen Parameter geändert haben, sollten Sie für die Anwendungen eventuell einen Rebind durchführen (mit dem Befehl **REBIND PACKAGE**).

comm_exit_list - Liste der Bibliotheken für Kommunikationspufferexits

Dieser Parameter gibt die Liste der Bibliotheken für Kommunikationspufferexits an, die DB2 verwenden wird. Eine Bibliothek für Kommunikationspufferexits ist eine dynamisch geladene Bibliothek, die von Anwendungen anderer Anbieter verwendet werden kann, um auf die zur Kommunikation mit Clientanwendungen verwendeten DB2-Kommunikationspuffer zuzugreifen und diese zu untersuchen.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
`Null [beliebige gültige Zeichenfolge]`

Der Wert ist standardmäßig null. Wenn der Wert nicht null ist, wird er als eine durch Kommas getrennte Liste von Bibliotheken für Kommunikationspuffer behandelt, die DB2 verwenden soll. Die einzelnen Bibliotheken müssen durch Kommas getrennt werden, ohne Leerzeichen vor oder nach dem Komma.

Jeder Name kann maximal 32 Byte lang sein. Der Name sollte nicht die Erweiterung enthalten (z. B. '.so' oder '.a'). Die Gesamtlänge des Parameters darf 128 Byte nicht überschreiten.

conn_elapse - Antwortzeit für Verbindung

Dieser Parameter gibt an, innerhalb welcher Anzahl Sekunden eine Netzverbindung zwischen DB2-Membern aufgebaut werden soll.

Konfigurationstyp
Datenbankmanager

Gilt für
DB2 pureScale-Server (mit mehreren DB2-Membern)

Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
10 [0–100]

Maßeinheit
Sekunden

Wird der Verbindungsaufbau innerhalb der durch den Parameter angegebenen Zeit erfolgreich durchgeführt, kommt es zum Datenaustausch. Wird die Verbindung nicht rechtzeitig aufgebaut, wird ein weiterer Versuch zum Verbindungsaufbau durchgeführt. Kommt innerhalb der im Parameter **max_connretries** angegebenen Anzahl von Neuversuchen keine Verbindung zustande, wird eine Fehlernachricht ausgegeben.

cpuspeed - CPU-Geschwindigkeit

Dieser Parameter gibt die CPU-Geschwindigkeit der Maschine(n) wieder, auf der/denen die Datenbank installiert ist.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Anweisungsgrenzwert

Standardwert [Bereich]
-1 [$1x10^{-10}$ — 1] Der Wert -1 gibt an, dass der Wert dieses Parameters gemäß den Ergebnissen eines Messprogramms neu festgelegt wird.

Maßeinheit
Millisekunden

Dieses Programm wird ausgeführt, wenn keine Vergleichsergebnisse verfügbar sind, wenn die Daten für IBM RS/6000 Modell 530H in der Datei nicht gefunden werden oder wenn die Daten für Ihr System in der Datei nicht gefunden werden.

Sie können diesen Wert explizit festlegen, um ein Modell einer Produktionsumgebung auf Ihrem Testsystem zu erstellen oder die Auswirkungen einer Hardwareaufrüstung zu bewerten. Wenn der Wert auf -1 gesetzt wird, wird **cpuspeed** erneut berechnet.

Empfehlung: Sie sollten diesen Parameter nur anpassen, wenn Sie ein Modell einer anderen Umgebung erstellen wollen.

Der Wert für die CPU-Geschwindigkeit wird vom Optimierungsprogramm bei der Bestimmung von Zugriffspfaden verwendet. Wenn Sie diesen Parameter geändert haben, sollten Sie für die Anwendungen eventuell einen Rebind durchführen (mit dem Befehl **REBIND PACKAGE**).

cur_commit - Zurzeit festgeschriebene Daten (Konfigurationsparameter)

Dieser Parameter steuert das Verhalten von Suchläufen mit der Isolationsstufe 'Cursorstabilität' (CS).

Konfigurationstyp
Datenbank

Parametertyp

- Konfigurierbar
- Nach Member in einer DB2 pureScale-Umgebung konfigurierbar

Standardwert [Bereich]
`ON [ON, AVAILABLE, DISABLED]`

Für neue Datenbanken ist der Standardwert auf `ON` gesetzt. Wenn der Standardwert auf `ON` gesetzt ist, gibt Ihre Abfrage die zurzeit festgeschriebenen Werte der Daten zum Zeitpunkt der Übergabe Ihrer Abfrage zurück.

Bei einem Datenbankupgrade von Version 9.5 oder einer früheren Version wird der Konfigurationsparameter **cur_commit** auf den Wert `DISABLED` gesetzt, um das Verhalten wie in früheren Releases beizubehalten. Wenn Sie mit zurzeit festgeschriebenen Daten bei Suchläufen mit Cursorstabilität arbeiten wollen, müssen Sie den Konfigurationsparameter **cur_commit** im Anschluss an das Upgrade auf `ON` setzen.

Sie können den Konfigurationsparameter **cur_commit** explizit auf den Wert `AVAILABLE` setzen. Wenn Sie diesen Parameter definiert haben, müssen Sie das Verhalten zum Abrufen zurzeit festgeschriebener Daten explizit anfordern, um die Ergebnisse zu erhalten, die zurzeit festgeschrieben sind.

Anmerkung: Die drei Registrierdatenbankvariablen **DB2_EVALUNCOMMITTED**, **DB2_SKIPDELETED** und **DB2_SKIPINSERTED** sind vom Parameter **cur_commit** betroffen, wenn die Isolationsstufe der Cursorstabilität verwendet wird. Diese Registrierdatenbankvariablen werden ignoriert, wenn die Option **USE CURRENTLY COMMITTED** oder **WAIT FOR OUTCOME** explizit beim Bindevorgang (**BIND**) oder bei der Anweisungsvorbereitung angegeben wird.

Anmerkung: Leistungsaspekte können in einer Datenbank eine Rolle spielen, in der signifikante Sperrenkonflikte bei der Verwendung von 'cur_commit' auftreten. Die festgeschriebene Version der Zeile wird aus dem Protokoll abgerufen, wobei eine bessere Leistung erzielt wird und Protokollplattenaktivitäten vermieden werden, wenn sich der Protokollsatz noch im Protokollpuffer befindet. Daher kann es empfehlenswert sein, den Wert für den Parameter **logbufsz** zu erhöhen, um die Leistung beim Abrufen zuvor festgeschriebener Daten zu verbessern.

date_compat - DATE-Kompatibilität (Datenbankkonfigurationsparameter)

Dieser Parameter gibt an, ob die DATE-Kompatibilitätssemantik, die mit dem Datentyp TIMESTAMP(0) verknüpft ist, auf die verbundene Datenbank angewendet wird.

Konfigurationstyp
Datenbank

Parametertyp
Informativ

Der Wert wird bei der Erstellung der Datenbank festgelegt und basiert auf der Einstellung der Registrierdatenbankvariablen **DB2_COMPATIBILITY_VECTOR** für den Datentyp DATE. Der Wert kann nicht geändert werden.

dft_account_str - Standardzeichenfolge für Abrechnung

Dieser Parameter fungiert als Standardsuffix für Abrechnungskennungen.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
NULL [beliebige gültige Zeichenfolge]

Bei jeder Verbindungsanforderung durch eine Anwendung wird eine Abrechnungszeichenfolge vom Anwendungsrequester an einen DRDA-Anwendungsserver gesendet, die aus einem von DB2 Connect erstellten Präfix und einem vom Benutzer eingegebenen Suffix besteht. Anhand dieser Abrechnungsdaten kann ein Systemadministrator die Ressourcennutzung für jeden Benutzerzugriff bestimmen.

Anmerkung: Dieser Parameter ist nur für DB2 Connect gültig.

Das Suffix wird von dem Anwendungsprogramm bereitgestellt, das die API sqlesact() aufruft, oder von dem Benutzer, der die Umgebungsvariable **DB2ACCOUNT** definiert. Wenn von der API oder der Umgebungsvariablen kein Suffix bereitgestellt wird, verwendet DB2 Connect den Wert dieses Parameters als Standardsuffix. Dieser Parameter ist insbesondere für Datenbankclients früherer Versionen (alle vor Version 2) nützlich, die nicht über Funktionen zum Senden einer Abrechnungszeichenfolge an DB2 Connect verfügen.

Empfehlung: Verwenden Sie für die Abrechnungszeichenfolge die folgenden zulässigen Werte:

- Alphabetische Zeichen (A - Z)
- Numerische Zeichen (0 - 9)
- Unterstreichungszeichen (_)

dft_monswitches - Monitorschalter des Standarddatenbanksystems

Dieser Parameter ermöglicht Ihnen, eine Reihe von Schaltern zu definieren, die intern jeweils als ein Bit des Parameters dargestellt werden.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Anmerkung: Die Änderung tritt unmittelbar in Kraft, wenn Sie explizit eine Verbindung zur Instanz herstellen, bevor Sie die Schaltereinstellungen für 'dft_mon_xxxx' ändern. Ansonsten tritt die Einstellung beim nächsten Neustart der Instanz in Kraft.

Standardwert
Alle Schalter ausgeschaltet außer dem standardmäßig eingeschalteten *dft_mon_timestamp*

Der Parameter ist dahingehend eindeutig, dass Sie jeden dieser Schalter unabhängig aktualisieren können, indem Sie die folgenden Parameter festlegen:

dft_mon_uow
Standardwert des UOW-Schalters (Unit of Work - Arbeitseinheit) von Snapshot Monitor

dft_mon_stmt
Standardwert des Schalters für SQL-Anweisungen von Snapshot Monitor

dft_mon_table
Standardwert des Schalters für Tabellen von Snapshot Monitor

dft_mon_bufpool
Standardwert des Schalters für Pufferpool von Snapshot Monitor

dft_mon_lock
Standardwert des Schalters für Sperren von Snapshot Monitor

dft_mon_sort
Standardwert des Schalters für Sortiervorgänge von Snapshot Monitor

dft_mon_timestamp
Standardwert des Schalters für die Zeitmarke des Überwachungsprogramms für Momentaufnahme

Empfehlung: Jeder auf ON gesetzte Schalter (außer dft_mon_timestamp) weist den Datenbankmanager an, die zu diesem Schalter gehörenden Monitordaten zu sammeln. Das Erfassen zusätzlicher Monitordaten erhöht die Verarbeitungszeit des Datenbankmanagers, wodurch die Systemleistung beeinträchtigt werden kann. Das Setzen des Schalters *dft_mon_timestamp* auf OFF wird wichtig, sobald sich die CPU-Auslastung 100% nähert. Wenn dies eintritt, wird die zur Ausgabe von Zeitmarken benötigte CPU-Zeit drastisch erhöht. Wenn der Zeitmarkenschalter inaktiviert ist, reduziert dies zudem erheblich den Gesamtaufwand für andere Daten in der Monitorschaltersteuerung.

Alle Überwachungsanwendungen erhalten diese Standardeinstellungen für die Schalter, wenn die Anwendung die erste Überwachungsanforderung absetzt (z. B. einen Schalter einstellt, den Ereignismonitor aktiviert, eine Momentaufnahme macht). In der Konfigurationsdatei sollten Sie einen Schalter nur dann aktivieren, wenn Sie Daten sammeln möchten, sobald der Datenbankmanager gestartet wird.

(Andernfalls kann jede Überwachungsanwendung ihre eigenen Schalter einstellen, und die gesammelten Daten beziehen sich dann auf den Zeitpunkt, zu dem die Schalter eingestellt wurden.)

dftdbpath - Standarddatenbankpfad

Dieser Parameter enthält den Standarddateipfad, der zur Erstellung von Datenbanken unter dem Datenbankmanager verwendet wird. Wird beim Erstellen einer Datenbank kein Pfad angegeben, wird die Datenbank in dem Pfad erstellt, der durch den Parameter **dftdbpath** angegeben wird.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]

UNIX Benutzerverzeichnis des Instanzeigners [beliebiger vorhandener Pfad]

Windows
Laufwerk, auf dem das DB2-Datenbanksystem installiert ist [beliebiger vorhandener Pfad]

In einer Umgebung mit partitionierten Datenbanken müssen Sie sicherstellen, dass der Pfad, in dem die Datenbank erstellt wird, kein an NFS angehängter Pfad (unter Linux- und UNIX-Betriebssystemen) bzw. kein Netzlaufwerk ist (in Windows-Umgebungen). Der angegebene Pfad muss physisch auf jedem Datenbankpartitionsserver vorhanden sein. Um Verwirrung zu vermeiden, ist es am besten, einen Pfad anzugeben, der auf jedem Datenbankpartitionsserver lokal angehängt ist. Die Länge des Pfads darf maximal 205 Zeichen betragen. Das System hängt den Namen der Datenbankpartition am Ende des Pfads an.

Weil Datenbanken auf beträchtliche Größen anwachsen und möglicherweise viele Benutzer Datenbanken erstellen können (je nach Umgebung und Zielsetzung), ist es häufig sehr praktisch, alle Datenbanken an einer einzigen definierten Position erstellen und speichern zu lassen. Außerdem ist es von Vorteil, Datenbanken von anderen Anwendungen und Daten trennen zu können - sowohl aus Integritätsgründen als auch zur einfacheren Durchführung von Backup- und Recovery-Operationen.

In Linux- und UNIX-Umgebungen darf die Länge des im Parameter **dftdbpath** definierten Namens 215 Zeichen nicht überschreiten, und der Name muss ein gültiger, absoluter Pfadname sein. Unter Windows kann der im Parameter **dftdbpath** angegebene Name ein Laufwerkbuchstabe sein, auf den optional ein Doppelpunkt folgt.

Empfehlung: Wenn die Möglichkeit besteht, legen Sie umfangreiche Datenbanken auf einer anderen Platte an als andere Daten, auf die häufig zugegriffen wird, wie z. B. Dateien des Betriebssystems oder die Datenbankprotokolldateien.

diaglevel - Aufzeichnungsebene bei Fehlerdiagnose

Durch diesen Parameter wird der Typ der Diagnosefehler festgelegt, die in der db2diag-Protokolldatei aufgezeichnet werden.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
3 [0 — 4]

Gültige Werte für diesen Parameter sind:

- **0** – Nur kritische Fehler, Ereignisnachrichten und Hinweisnachrichten für die Systemverwaltung werden serverseitig erfasst. Als Wert für diesen Parameter darf auf der Clientseite nicht `0` festgelegt werden.
- **1** – Nur schwerwiegende Fehler, kritische Fehler, Ereignisnachrichten und Hinweisnachrichten für die Systemverwaltung werden erfasst.
- **2** – Alle Fehler, Ereignisnachrichten und Hinweisnachrichten für die Systemverwaltung werden erfasst.
- **3** – Alle Fehler, Warnungen, Ereignisnachrichten und Hinweisnachrichten für die Systemverwaltung werden erfasst.
- **4** – Alle Fehler, Warnungen, Informationsnachrichten, Ereignisnachrichten und Hinweisnachrichten für Systemverwaltung werden erfasst.

Der Konfigurationsparameter **diagpath** wird zur Angabe des Verzeichnisses verwendet, in dem sich die Fehlerdatei, die Alertprotokolldatei und alle Speicherauszugsdateien befinden, die in Abhängigkeit vom Wert des Parameters **diaglevel** generiert werden.

Hinweise

- Das dynamische Verhalten für **diaglevel** erstreckt sich nicht auf alle Prozesse.
- Der DB2-Serverprozess **db2sysc** kann dynamische Änderungen erkennen, z. B. wenn der Befehl **UPDATE DATABASE MANAGER CONFIGURATION** für eine Instanzzuordnung ausgegeben wird.
- Wenn DB2-Client- und -Anwendungsprozesse gestartet werden, wird dafür die Einstellung des Konfigurationsparameters **diaglevel** verwendet und es werden keine dynamischen Änderungen erkannt.
- Der Wert dieses Parameters kann erhöht werden, um zusätzliche Fehlerbestimmungsdaten zu sammeln, die zur Lösung eines Problems beitragen können.

diagpath - Verzeichnispfad für Diagnosedaten

Mit diesem Parameter können Sie den vollständig qualifizierten Primärpfad für DB2-Diagnosedaten angeben.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
$*INSTHOME*/sqllib/db2dump/ $m [beliebiger gültiger Pfadname (Optionen für die Auflösung von Variablen siehe nachfolgender Abschnitt)]

Symbole

pfadname
Ein Verzeichnispfad, der anstelle des Standardverzeichnisses für Diagnosedaten verwendet wird.

$h Wird in HOST_*hostname* aufgelöst.

Anmerkung: Ab Version 10 bezieht sich $h in DB2 pureScale-Umgebungen auf den Benutzerhost des Members.

$n Wird in NODE*nummer* aufgelöst.

$m Wird in DIAG*nummer* aufgelöst. Dabei ist zu beachten, dass die Verwendung von DIAG*nummer* nicht davon abhängig ist, ob dieser Wert auf eine Datenbankpartition verweist, eine CF oder auf ein Member.

/anschließendes_verzeichnis
Ein einzelnes Verzeichnis oder ein Verzeichnis mit Unterverzeichnis, das sich an die Angabe $h oder $n anschließt.

Die folgenden Werte sind verfügbar:

- '"$h"'
- '"$h/*anschließendes_verzeichnis*"'
- '"*pfadname* $h"'
- '"*pfadname* $h/*anschließendes_verzeichnis*"'
- '"$n"'[4]
- '"$n/*anschließendes_verzeichnis*"'
- '"*pfadname* $n"'
- '"*pfadname* $n/*anschließendes_verzeichnis*"'
- '"$m"'

4. $n ist veraltet und wird in einem zukünftigen Release möglicherweise entfernt.

- '"$m/*anschließendes_verzeichnis*"'
- '"*pfadname* $m"'
- '"*pfadname* $m/*anschließendes_verzeichnis*"'
- '"hn"'[5]
- '"hn/*anschließendes_verzeichnis*"'
- '"*pfadname* hn"'
- '"*pfadname* hn/*anschließendes_verzeichnis*"'
- '"hm"'
- '"hm/*anschließendes_verzeichnis*"'
- '"*pfadname* hm"'
- '"*pfadname* hm/*anschließendes_verzeichnis*"'

Im primären Verzeichnis für Diagnosedaten können abhängig von Ihrer Plattform Speicherauszugsdateien, Trapdateien, eine Fehlerprotokolldatei, eine Benachrichtigungsdatei, eine Alertprotokolldatei und FODC-Pakete (FODC = First Occurrence Date Collection, Datenerfassung beim ersten Vorkommen) gespeichert werden.

Wenn dieser Parameter den Wert 'null' hat, werden die Diagnoseinformationen in eine Zeichenfolge in einem der folgenden Verzeichnisse (bzw. Ordner) im Standarddiagnosepfad geschrieben:

- In Windows-Umgebungen:
  - Die Standardposition von Benutzerdatendateien, zum Beispiel Dateien unter Instanzverzeichnissen, variiert je nach Edition der Windows-Betriebssystemfamilie. Verwenden Sie den Befehl **DB2SET DB2INSTPROF**, um die Position des Instanzverzeichnisses abzurufen. Doe Datei befindet sich im Unterverzeichnis `instance` des Verzeichnisses, das durch die Registrierdatenbankvariable **DB2INSTPROF** angegeben wird.
- In Linux- und UNIX-Umgebungen: Informationen werden in das Verzeichnis *INSTHOME*`/sqllib/db2dump/ $m` geschrieben. Dabei ist *INSTHOME* das Ausgangsverzeichnis der Instanz.

Ab Version 10.1 schreibt der Verzeichnispfad für Diagnosedaten für jedes Member und jede CF standardmäßig in die private Datei 'db2diag.log'. Zum Zurücksetzen auf das Verhalten der früheren Releases, in denen die Diagnosedaten in dasselbe Verzeichnis geschrieben wurden, geben Sie **diagpath** mit einem *pfadnamen* und ohne Token (`$h`, `$n` oder `$m`) an.

Um den Verzeichnispfad für Diagnosedaten zu teilen, sodass die Diagnoseinformationen pro physischen Host getrennt erfasst werden, setzen Sie den Parameter auf einen der folgenden Werte:

- Teilen Sie den Standardverzeichnispfad für Diagnosedaten:
  ```
  db2 update dbm cfg using diagpath '"$h"'
  ```
  Dadurch wird ein Unterverzeichnis unter dem Standardverzeichnis für Diagnosedaten mit dem Hostnamen erstellt, wie aus folgendem Beispiel hervorgeht:
  standard-diagpfad`/HOST_`*hostname*
- Teilen Sie den Standardverzeichnispfad für Diagnosedaten mit einem anschließenden Verzeichnis:
  ```
  db2 update dbm cfg using diagpath '"$h/anschließendes_verzeichnis"'
  ```

5. hn ist veraltet und wird in einem zukünftigen Release möglicherweise entfernt.

Dadurch wird ein Unterverzeichnis unter dem Standardverzeichnis für Diagnosedaten mit dem Hostnamen und einem anschließenden Verzeichnis erstellt, wie aus folgendem Beispiel hervorgeht:

standard-diagpfad/HOST_*hostname*/*anschließendes_verzeichnis*

- Teilen Sie Ihren selbstdefinierten Verzeichnispfad für Diagnosedaten (zwischen *pfadname* und $h befindet sich ein Leerzeichen):

  ```
  db2 update dbm cfg using diagpath '"pfadname $h"'
  ```

  Dadurch wird ein Unterverzeichnis unter Ihrem selbstdefinierten Verzeichnis für Diagnosedaten mit dem Hostnamen erstellt, wie aus folgendem Beispiel hervorgeht:

  pfadname/HOST_*hostname*

- Teilen Sie Ihren selbstdefinierten Verzeichnispfad für Diagnosedaten (zwischen *pfadname* und $h befindet sich ein Leerzeichen) mit einem anschließenden Verzeichnis:

  ```
  db2 update dbm cfg using diagpath '"pfadname $h/anschließendes_verzeichnis"'
  ```

  Dadurch wird ein Unterverzeichnis unter Ihrem selbstdefinierten Verzeichnis für Diagnosedaten mit dem Hostnamen und einem anschließenden Verzeichnis erstellt, wie aus folgendem Beispiel hervorgeht:

  pfadname/HOST_*hostname*/*anschließendes_verzeichnis*

Um den Verzeichnispfad für Diagnosedaten zu teilen, sodass die Diagnoseinformationen pro physischen Host und pro Datenbankpartition pro physischen Host getrennt erfasst werden, setzen Sie den Parameter auf einen der folgenden Werte:

- Teilen Sie den Standardverzeichnispfad für Diagnosedaten:

  ```
  db2 update dbm cfg using diagpath '"$h$n"'
  ```

  Dadurch wird ein Unterverzeichnis für jede logische Partition auf dem Host unter dem Standardverzeichnis für Diagnosedaten mit dem Hostnamen und der Partitionsnummer erstellt, wie aus folgendem Beispiel hervorgeht:

  standard-diagpfad/HOST_*hostname*/NODE*nummer*

- Teilen Sie den Standardverzeichnispfad für Diagnosedaten mit einem anschließenden Verzeichnis:

  ```
  db2 update dbm cfg using diagpath '"$h$n/anschließendes_verzeichnis"'
  ```

  Dadurch wird ein Unterverzeichnis für jede logische Partition auf dem Host unter dem Standardverzeichnis für Diagnosedaten mit dem Hostnamen, der Partitionsnummer und einem anschließenden Verzeichnis erstellt, wie aus folgendem Beispiel hervorgeht:

  standard-diagpfad/HOST_*hostname*/NODE*nummer*/*anschließendes_verzeichnis*

- Teilen Sie Ihren selbstdefinierten Verzeichnispfad für Diagnosedaten (zwischen *pfadname* und hn befindet sich ein Leerzeichen):

  ```
  db2 update dbm cfg using diagpath '"pfadname $h$n"'
  ```

  Dadurch wird ein Unterverzeichnis für jede logische Partition auf dem Host unter dem selbstdefinierten Verzeichnis für Diagnosedaten mit dem Hostnamen und der Partitionsnummer erstellt, wie aus folgendem Beispiel hervorgeht:

  pfadname/HOST_*hostname*/NODE*nummer*

  Beispiel: Ein AIX-Host mit dem Namen boson hat drei Datenbankpartitionen mit den Knotennummern 0, 1 und 2. Eine Listenausgabe des Verzeichnisses könnte z. B. wie folgt aussehen:

```
usr1@boson /home/user1/db2dump->ls -R *
HOST_boson:

HOST_boson:
NODE0000 NODE0001 NODE0002

HOST_boson/NODE0000:
db2diag.log db2eventlog.000 db2resync.log db2sampl_Import.msg events usr1.nfy

HOST_boson/NODE0000/events:
db2optstats.0.log

HOST_boson/NODE0001:
db2diag.log db2eventlog.001 db2resync.log usr1.nfy stmmlog

HOST_boson/NODE0001/stmmlog:
stmm.0.log

HOST_boson/NODE0002:
db2diag.log db2eventlog.002 db2resync.log usr1.nfy
```

- Teilen Sie Ihren selbstdefinierten Verzeichnispfad für Diagnosedaten (zwischen *pfadname* und hn befindet sich ein Leerzeichen) mit einem anschließenden Verzeichnis:

  ```
  db2 update dbm cfg using diagpath '"pfadname $h$n/anschließendes_verzeichnis"'
  ```

  Dadurch wird ein Unterverzeichnis für jede logische Partition auf dem Host unter dem selbstdefinierten Verzeichnis für Diagnosedaten mit dem Hostnamen, der Partitionsnummer und einem anschließenden Verzeichnis erstellt, wie aus folgendem Beispiel hervorgeht:

  pfadname/HOST_*hostname*/NODE*nummer*/*anschließendes_verzeichnis*

Anmerkung:

- Damit die Betriebssystemshell das Zeichen $ auf Linux- und UNIX-Systemen nicht als Steuerzeichen interpretiert, müssen außerhalb der doppelten Anführungszeichen zusätzlich einfache Anführungszeichen (Hochkommas) eingegeben werden, wie aus dem Syntaxbeispiel hervorgeht.
- Im CLP-Dialogmodus - oder wenn der Befehl aus einer Eingabedatei gelesen und ausgeführt wird - ist das doppelte Anführungszeichen nicht erforderlich.
- Bei $h, $n und $m wird nicht zwischen Groß- und Kleinschreibung unterschieden.
- Das dynamische Verhalten für **diagpath** erstreckt sich nicht auf alle Prozesse.
- Der DB2-Serverprozess **db2sysc** kann dynamische Änderungen erkennen, z. B. wenn der Befehl **UPDATE DATABASE MANAGER CONFIGURATION** für eine Instanzzuordnung ausgegeben wird.
- Wenn DB2-Client- und -Anwendungsprozesse gestartet werden, wird dafür die Einstellung des Konfigurationsparameters **diagpath** verwendet und es werden keine dynamischen Änderungen erkannt.

In Version 9.5 und späteren Versionen wird der Standardwert von **DB2INSTPROF** auf der globalen Ebene an der zuvor gezeigten neuen Position gespeichert. Andere Profilregistrierdatenbankvariablen, die die Position der Laufzeitdatendateien angeben, müssen den Wert von **DB2INSTPROF** abfragen. Bei den anderen Variablen handelt es sich um die folgenden:

- **DB2CLIINIPATH**
- **DIAGPATH**
- **SPM_LOG_PATH**

Anmerkung: In DB2 Version 9.7 Fixpack 4 und in höheren Fixpacks kann die Protokollierung von Diagnosedaten durch Definieren eines alternativen Diagnosepfads zusammen mit dem Parameter **diagpath** mit erhöhter Ausfallsicherheit ausgestattet werden. Wenn der Parameter **alt_diagpath** definiert wird und der im Parameter **diagpath** angegebene Pfad nicht verfügbar ist, läuft die Protokollierung von Diagnosedaten im angegebenen alternativen Verzeichnispfad für Diagnosedaten weiter und wird fortgesetzt, wenn der primäre Diagnosepfad wieder verfügbar ist.

diagsize - Rollierende Protokolle mit Benachrichtigungen für die Systemverwaltung und für die Diagnose (Konfigurationsparameter)

Dieser Parameter hilft bei der Steuerung der Maximalgrößen der Dateien für das Diagnoseprotokoll und das Protokoll mit Benachrichtigungen für die Systemverwaltung.

Konfigurationstyp
Datenbankmanager

Parametertyp
Nicht online konfigurierbar

Standardwert
0

Der Mindestwert zur Angabe der Größe rollierender Protokolle:
2

Der Maximalwert zur Angabe der Größe rollierender Protokolle:
Der Betrag an freiem Speicherbereich in dem Verzeichnis, das durch 'diagpath' angegeben wird

Maßeinheit
Megabyte

Wenn dieser Parameter den Wert `0` (Standardwert) hat, ist nur eine Diagnoseprotokolldatei mit dem Namen `db2diag.log` vorhanden. Es gibt zudem nur eine Protokolldatei mit Benachrichtigungen für die Systemverwaltung mit dem Namen `<instanz>.nfy`, die nur unter Linux- und UNIX-Betriebssystemen verwendet wird. Die Größen dieser Dateien können unbegrenzt ansteigen.

Wenn Sie den Parameter auf einen Wert ungleich null setzen und die Instanz (<instanz>) erneut starten, wird eine Reihe von rollierenden Diagnoseprotokolldateien und eine Reihe von rollierenden Protokolldateien mit Benachrichtigungen für die Systemverwaltung verwendet. Diese Dateien haben das Benennungsformat `db2diag.n.log` und `<instanz>.n.nfy`, wobei *n* eine ganzzahlige Nummer ist. Dateien mit Namen im Format `<instanz>.n.nfy` werden nur unter Linux- und UNIX-Betriebssystemen verwendet. Die Nummer in Dateien `db2diag.n.log` und `<instanz>.n.nfy` kann den Wert 10 nicht überschreiten. Wenn die Größe der zehnten Datei ausgeschöpft ist, wird die älteste Datei gelöscht und eine neue Datei erstellt.

Unter Linux- und UNIX-Betriebssystemen könnten die rollierenden Protokolldateien unter **diagpath** wie bei der folgenden Beispielausgabe aussehen:

```
db2diag.14.log, db2diag.15.log, ... , db2diag.22.log, db2diag.23.log
<instanz>.0.nfy, <instanz>.1.nfy...,<instanz>.8.nfy,<instanz>.9.nfy
```

Wenn die Datei 'db2diag.23.log' voll ist, wird die Datei 'db2diag.14.log' gelöscht und die Datei 'db2diag.24.log' wird für die db2diag-Protokollierung erstellt.

Wenn die Datei <instanz>.9.nfy voll ist, wird die Datei <instanz>.0.nfy gelöscht und die Datei <instanz>.10.nfy wird für die Protokollierung mit Benachrichtigungen für die Systemverwaltung erstellt.

Beachten Sie, dass die Nachrichten stets in der rollierenden Protokolldatei mit der höchsten Indexnummer (d. h. db2diag.*höchstes n*.log, <*instanz*>.*höchstes n*.nfy) protokolliert werden.

Die Gesamtgröße der Dateien db2diag.*n*.log und <*instanz*>.*n*.nfy wird durch den Wert des Konfigurationsparameters **diagsize** festgelegt. Standardmäßig, außer unter Windows-Betriebssystemen, werden 90 % des Werts von **diagsize** den Dateien db2diag.*n*.log und 10 % des Werts von **diagsize** den Dateien <*instanz*>.*n*.nfy zugeordnet. Wenn Sie den Parameter **diagsize** unter einem Linux- oder UNIX-Betriebssystem zum Beispiel auf den Wert 1024 setzen, kann die Gesamtgröße der Dateien db2diag.*n*.log den Wert 921,6 MB und die Gesamtgröße der Dateien <*instanz*>.*n*.nfy den Wert 102,4 MB nicht überschreiten. Unter Windows-Betriebssystemen wird der Wert von **diagsize** insgesamt den Dateien db2diag.*n*.log zugeordnet. Die Größe jeder Protokolldatei entspricht der Gesamtkapazität an Speicherbereich, die jedem Typ von Protokolldatei zugeordnet wird, dividiert durch 10. Wenn zum Beispiel die Gesamtgröße der Dateien db2diag.*n*.log den Wert 921,6 MB nicht überschreiten kann, beträgt die Größe jeder einzelnen Datei db2diag.*n*.log 92,16 MB.

Der Maximalwert, den Sie für den Konfigurationsparameter **diagsize** angeben können, darf die Kapazität an freiem Speicherbereich in dem Verzeichnis, das Sie im Konfigurationsparameter **diagpath** angeben, nicht überschreiten. Die Dateien für das Diagnoseprotokoll und das Protokoll mit Benachrichtigungen für die Systemverwaltung werden in diesem Verzeichnis gespeichert. Um zu vermeiden, dass Informationen aufgrund des Dateiumlaufs (d. h. durch das Löschen der ältesten Protokolldatei) zu schnell verloren gehen, setzen Sie den Parameter **diagsize** auf einen Wert größer als 50 MB, jedoch nicht größer als 80 % des freien Speicherbereichs in dem Verzeichnis, das Sie in **diagpath** angeben.

Beispiel:

- Wenn Sie **diagsize** auf den Wert 1024 MB setzen wollen, sodass zum rollierenden Protokollverhalten gewechselt wird, wenn DB2 erneut gestartet wird, verwenden Sie den folgenden Befehl:

  ```
  db2 update dbm cfg using diagsize 1024
  ```

- Wenn Sie **diagsize** auf den Wert 0 setzen wollen, sodass zum Standardprotokollverhalten gewechselt wird, wenn DB2 erneut gestartet wird, verwenden Sie den folgenden Befehl:

  ```
  db2 update dbm cfg using diagsize 0
  ```

Anmerkung: Ab DB2 Version 9.7 Fixpack 1 gilt: Wenn der Konfigurationsparameter **diagsize** auf einen Wert ungleich null gesetzt wird und der Konfigurationsparameter **diagpath** so gesetzt wird, dass die Diagnosedaten auf separate Verzeichnisse aufgeteilt werden, dann gibt der Wert ungleich null des Konfigurationsparameters **diagsize** die Gesamtgröße aller rotierenden Protokolldateien mit Benachrichtigungen für die Systemverwaltung zusammen mit allen rotierenden Protokollen der Diagnoseprogramme innerhalb eines jeweiligen geteilten Diagnosedatenverzeichnisses an. Beispiel: Wenn in einem System mit 4 Datenbankpartitionen der Parameter **diagsize** auf 1 GB und der Parameter **diagpath** auf "$n"

(Diagnosedaten pro Datenbankpartition aufteilen) gesetzt ist, dann kann die Gesamtgröße der Benachrichtigungsprotokolle und der Protokolle der Diagnoseprogramme maximal 4 GB (4 x 1 GB) betragen.

dir_cache - Verzeichniscacheunterstützung

Dieser Parameter legt fest, ob die Datenbank-, die Knoten- und die DCS-Verzeichnisdateien in den Cache gestellt werden.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
`Yes [Yes; No ]`

Zuordnung

- Wenn eine Anwendung die erste Verbindungsanforderung absetzt, wird der Anwendungsverzeichniscache zugeordnet.
- Wenn eine Datenbankmanagerinstanz gestartet (**db2start**) wird, wird der Serververzeichniscache zugeordnet.

Freigabe

- Wenn ein Anwendungsprozess beendet wird, wird der Anwendungsverzeichniscache freigegeben.
- Wenn eine Datenbankmanagerinstanz gestoppt (**db2stop**) wird, wird der Serververzeichniscache freigegeben.

Durch die Verwendung des Verzeichniscache wird der Aufwand für die Verbindung verringert, indem die E/A-Operationen für Verzeichnisdateien vermieden und die Suchoperationen in Verzeichnissen zum Abruf von Verzeichnisdaten minimiert werden. Es gibt zwei Arten von Verzeichniscaches:

- Ein Anwendungsverzeichniscache, der für jeden Anwendungsprozess auf dem System zugeordnet und verwendet wird, auf dem die Anwendung ausgeführt wird.
- Ein Serververzeichniscache, der für einige der internen Datenbankmanagerprozesse zugeordnet und verwendet wird.

Wenn eine Anwendung die erste Verbindungsanforderung absetzt, werden in einem Anwendungsverzeichniscache alle Verzeichnisdateien gelesen, und die Informationen werden im privaten Speicher dieser Anwendung zwischengespeichert. Der Cache wird vom Anwendungsprozess auch für nachfolgende Verbindungsanforderungen verwendet und während der Dauer des Anwendungsprozesses beibehalten. Wenn eine Datenbank nicht im Anwendungsverzeichniscache gefunden wird, werden die Verzeichnisdateien nach den Informationen durchsucht, aber der Cache wird nicht aktualisiert. Wenn die Anwendung einen Verzeichniseintrag ändert, wird durch die nächste Verbindungsanforderung innerhalb dieser Anwendung eine Aktualisierung des Cache für diese Anwendung bewirkt. Der Anwendungsverzeichniscache für andere Anwendungen wird nicht aktualisiert. Wenn der

Anwendungsprozess beendet ist, wird der Cache freigegeben. (Zur Aktualisierung des Verzeichniscache, der von einer Sitzung des Befehlszeilenprozessors verwendet wird, geben Sie den Befehl **db2 terminate** ein.)

Wenn eine Datenbankmanagerinstanz gestartet wird (**db2start**), werden in einem Serververzeichniscache alle Verzeichnisdateien gelesen, und die Informationen werden im Servercache zwischengespeichert. Dieser Cache wird beibehalten, bis die Instanz gestoppt (**db2stop**) wird. Wenn ein Verzeichniseintrag in diesem Cache nicht gefunden wird, werden die Verzeichnisdateien nach den Daten durchsucht. Normalerweise wird dieser Serververzeichniscache während der Ausführung der Instanz zu keinem Zeitpunkt aktualisiert. Durch ein Offline-Backup wird der Serververzeichniscache allerdings als ungültig markiert, wodurch er aktualisiert wird, auch wenn die Instanz gerade ausgeführt wird.

Empfehlung: Verwenden Sie einen Verzeichniscache, wenn Ihre Verzeichnisdateien nicht häufig geändert werden und die Leistung von entscheidender Bedeutung ist.

Auf fernen Clients kann darüber hinaus der Verzeichniscache von Vorteil sein, wenn Ihre Anwendungen mehrere verschiedene Verbindungsanforderungen absetzen. In diesem Fall verringert das Zwischenspeichern die Häufigkeit, mit der eine einzige Anwendung die Verzeichnisdateien lesen muss.

Der Verzeichniscache kann auch die Leistung bei der Erstellung von Momentaufnahmen des Datenbanksystemmonitors erhöhen. Außerdem sollten Sie beim Aufruf der Momentaufnahme den Datenbanknamen explizit angeben und nicht den Aliasnamen für die Datenbank verwenden.

Anmerkung: Bei der Erstellung von Momentaufnahmen können Fehler auftreten, wenn der Verzeichniscache aktiviert wurde und Datenbanken katalogisiert, entkatalogisiert, erstellt oder gelöscht werden, nachdem der Datenbankmanager gestartet wurde.

discover - Discovery-Modus

Mit diesem Parameter wird ermittelt, welche Art von Erkennungsanforderungen der Client (falls überhaupt) vornehmen kann.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
`SEARCH [DISABLE, KNOWN, SEARCH]`

Aus der Perspektive eines Clients gilt einer der folgenden Fälle:

- Wenn **SEARCH** für `discover` angegeben ist, kann der Client Discovery-Anforderungen mit dem Parameter SEARCH zum Suchen nach DB2-Serversystemen im Netz absetzen. Discovery mit dem Parameter SEARCH stellt eine Implikation der durch Discovery mit dem Parameter KNOWN bereitgestellten Funktionalität

zur Verfügung. Wenn **SEARCH** für discover angegeben ist, können vom Client sowohl Discovery-Anforderungen mit dem Parameter SEARCH als auch Discovery-Anforderungen mit dem Parameter KNOWN abgesetzt werden.

- Wenn für **discover** KNOWN angegeben ist, können vom Client nur Discovery-Anforderungen mit dem Parameter KNOWN abgesetzt werden. Durch Angabe einiger Verbindungsinformationen für den Verwaltungsserver auf einem bestimmten System werden alle Instanz- und Datenbankinformationen auf dem DB2-System an den Client zurückgegeben.
- Wenn für **discover** DISABLE angegeben ist, ist Discovery auf dem Client inaktiviert.

Der Standarderkennungsmodus ist SEARCH.

discover_inst - Discovery-Serverinstanz

Dieser Parameter gibt an, ob diese Instanz von DB2 Discovery aufgespürt werden kann.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
ENABLE [ENABLE, DISABLE]

Der Standardwert ENABLE für den Parameter gibt an, dass die Instanz erkannt werden kann, während DISABLE verhindert, dass die Instanz erkannt wird.

fcm_num_buffers - Anzahl FCM-Puffer

Dieser Parameter gibt die Anzahl von 4-KB-Puffern an, die sowohl zwischen den als auch innerhalb der Datenbankserver für interne Kommunikation (Nachrichten) verwendet werden.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver oder DB2 pureScale-Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]

32-Bit-Plattformen
`Automatic` [895 - 65300]

64-Bit-Plattformen
`Automatic` [895 - 524288]

- Datenbankserver mit lokalen und fernen Clients: 1024
- Datenbankserver mit lokalen Clients: 895
- Partitionierter Datenbankserver oder DB2 pureScale-Datenbankserver mit lokalen oder fernen Clients: 4096

Fast Communication Manager-Puffer (FCM) werden standardmäßig sowohl für die Kommunikation zwischen Membern als auch innerhalb eines Members verwendet.

Wichtig: Der Standardwert des Parameters **fcm_num_buffers** unterliegt nach der ursprünglichen Datenbankerstellung Änderungen durch den DB2 Configuration Advisor.

Sie können sowohl einen Anfangswert als auch den Wert `AUTOMATIC` für den Konfigurationsparameter **fcm_num_buffers** festlegen. Wenn Sie für den Parameter den Wert `AUTOMATIC` festlegen, überwacht FCM die Ressourcennutzung und kann Ressourcen erhöhen bzw. verringern, wenn sie innerhalb von 30 Minuten nicht genutzt werden. Der Wert, um den Ressourcen erhöht oder verringert werden, hängt vom Betriebssystem ab. Unter Linux-Betriebssystemen kann die Anzahl der Puffer gegenüber dem Anfangswert nur um 25 Prozent erhöht werden. Wenn der Datenbankmanager versucht, eine Instanz zu starten und die angegebene Anzahl Puffer nicht zuordnen kann, wird diese Anzahl verringert, bis die Instanz gestartet werden kann.

Falls für den Parameter **fcm_num_buffers** sowohl ein bestimmter Wert als auch `AUTOMATIC` festgelegt werden soll und Sie nicht möchten, dass der Systemcontroller-Thread die Ressourcen bis auf einen Wert unter dem angegebenen Wert anpasst, müssen Sie die Option `FCM_CFG_BASE_AS_FLOOR` der Registrierdatenbankvariable **DB2_FCM_SETTINGS** auf `YES` oder `TRUE` festlegen. Der Wert der Registrierdatenbankvariablen **DB2_FCM_SETTINGS** wird dynamisch angepasst.

Wenn Sie mehrere logische Knoten verwenden, wird ein Pool mit der im Parameter **fcm_num_buffers** angegebenen Anzahl von Puffern von allen logischen Knoten auf derselben Maschine gemeinsam genutzt. Sie können die Größe des Pools festlegen, indem Sie den Wert des Parameters **fcm_num_buffers** mit der Anzahl der logischen Knoten auf der physischen Maschine multiplizieren. Überprüfen Sie den Wert, den Sie verwenden. Berücksichtigen Sie, wie viele FCM-Puffer auf einer Maschine oder Maschinen mit mehreren logischen Knoten zugeordnet werden. Wenn Sie über mehrere logische Knoten auf derselben Maschine verfügen, müssen Sie den Wert des Parameters **fcm_num_buffers** möglicherweise erhöhen. Die Anzahl der Benutzer im System, die Anzahl der Datenbankpartitionsserver im System oder die Komplexität der Anwendungen kann dazu führen, dass die Anzahl der Nachrichtenpuffer in einem System nicht ausreicht.

fcm_num_channels - Anzahl FCM-Kanäle

Mit diesem Parameter wird die Anzahl der FCM-Kanäle für jede Datenbankpartition angegeben.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver oder DB2 pureScale-Datenbankserver mit lokalen und fernen Clients
- Satellitendatenbankserver mit lokalen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]

UNIX-Plattformen (32-Bit)
`Automatic` mit dem Anfangswert `256`, `512` oder `2048` [`128 - 120000`]

UNIX-Plattformen (64 Bit)
`Automatic` mit dem Anfangswert `256`, `512` oder `2048` [`128 - 524288`]

Windows (32 Bit)
`Automatic` mit dem Anfangswert `10000` [`128 - 120000`]

Windows (64 Bit)
`Automatic` mit dem Anfangswert `256`, `512` oder `2048` [`128 - 524288`]

Im Folgenden finden Sie die Standardanfangswerte für die verschiedenen Servertypen:

- Für einen Datenbankserver mit lokalen und fernen Clients ist der Anfangswert `512`.
- Für einen Datenbankserver mit lokalen Clients ist der Anfangswert `256`.
- Für Server in Umgebungen mit partitionierten Datenbanken mit lokalen und fernen Clients ist der Anfangswert `2048`.

Fast Communication Manager-Puffer (FCM) werden standardmäßig sowohl für die Kommunikation zwischen Membern als auch innerhalb eines Members verwendet. Damit Datenbanksysteme ohne Cluster das FCM-Subsystem und den Parameter **`fcm_num_channels`** verwenden können, musste der Parameter **`intra_parallel`** auf die Einstellung `YES` gesetzt werden.

Ein FCM-Kanal stellt einen logischen Kommunikationsendpunkt zwischen EDUs (Engine Dispatchable Units) dar, die in der DB2-Steuerkomponente ausgeführt werden. Sowohl Steuerungsflüsse (Anforderung und Antwort) als auch Datenflüsse (Daten aus Tabellenwarteschlangen) nutzen Kanäle, um Daten zwischen Membern zu übertragen.

Wenn dieser Parameter auf `AUTOMATIC` gesetzt ist, überwacht FCM die Kanalnutzung und ordnet Ressourcen je nach Bedarfsänderung inkrementell zu bzw. gibt sie inkrementell frei.

fed_noauth - Authentifizierung bei Servern mit föderierten Datenbanken umgehen

Dieser Parameter legt fest, ob die Authentifizierung für einen Server mit föderierten Datenbanken in der Instanz umgangen wird.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
`No [Yes; No]`

Wenn der Parameter **fed_noauth** auf `yes`, der Parameter **authentication** auf `server` oder auf `server_encrypt` und der Parameter **federated** auf `yes` gesetzt ist, wird die Authentifizierung in der Instanz umgangen. In diesem Fall wird angenommen, dass die Authentifizierung an der Datenquelle erfolgt. Gehen Sie vorsichtig vor, wenn **fed_noauth** auf `yes` gesetzt ist. Es wird weder auf dem Client noch in DB2 eine Authentifizierung durchgeführt. Jeder Benutzer, der den SYSADM-Authentifizierungsnamen kennt, kann SYSADM-Berechtigung für den Server mit föderierten Datenbanken erlangen.

federated - Unterstützung für Systeme föderierter Datenbanken

Dieser Parameter aktiviert bzw. inaktiviert die Unterstützung für Anwendungen, die verteilte Anforderungen für von Datenquellen (wie z. B. der DB2-Produktfamilie und Oracle) verwaltete Daten übergeben.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
No [Yes; No]

federated_async - Maximale ATQs pro Abfrage (Konfigurationsparameter)

Dieser Parameter legt die maximale Anzahl von ATQs (Asynchrony Table Queues - Asynchrony-Tabellenwarteschlangen) im Zugriffsplan fest, die vom Server mit föderierten Datenbanken unterstützt wird. Der ATQ-Mechanismus funktioniert in DB2 pureScale-Umgebungen nicht.

Konfigurationstyp
Datenbankmanager

Gilt für

- Partitionierten Datenbankserver mit lokalen und fernen Clients, wenn die Föderation aktiviert ist.

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
`0` [`0` bis `32 767` einschließlich, `-1`, `ANY`]

Wenn `ANY` oder `-1` angegeben ist, ermittelt das Optimierungsprogramm die Anzahl der ATQs für den Zugriffsplan. Das Optimierungsprogramm ordnet allen auswählbaren SHIP-Operatoren oder fernen Pushdown-Operatoren im Plan eine ATQ zu. Der für die Serveroption DB2_MAX_ASYNC_REQUESTS_PER_QUERY angegebene Wert schränkt die Anzahl der asynchronen Anforderungen ein.

Empfehlung
Der Konfigurationsparameter **federated_async** stellt den Standard- oder Anfangswert für das Sonderregister und die Bindeoption bereit. Sie können den Wert für diesen Parameter überschreiben, indem Sie den Wert des Sonderregisters CURRENT FEDERATED ASYNCHRONY, der Bindeoption **FEDERATED_ASYNCHRONY** oder der Vorkompilierungsoption **FEDERATED_ASYNCHRONY** erhöhen oder verringern.

Wenn das Sonderregister oder die Bindeoption den Konfigurationsparameter **federated_async** nicht überschreibt, legt der Wert des Parameters die maximale Anzahl von ATQs im Zugriffsplan fest, die der Server mit föderierten Datenbanken zulässt. Wenn das Sonderregister oder die Bindeoption diesen Parameter überschreibt, legt der Wert des Sonderregisters oder der Bindeoption die maximale Anzahl der ATQs im Plan fest.

Änderungen am Konfigurationsparameter **federated_async** wirken sich auf dynamische Anweisungen aus, sobald die aktuelle Arbeitseinheit festgeschrieben wird. Nachfolgende dynamische Anweisungen können den neuen Wert automatisch identifizieren. Ein Neustart der föderierten Datenbank ist nicht erforderlich. Pakete mit eingebettetem SQL werden weder inaktiviert noch implizit erneut gebunden, wenn der Wert des Konfigurationsparameters **federated_async** geändert wird.

Wenn Sie möchten, dass sich der neue Wert des Konfigurationsparameters **federated_async** auch auf statische SQL-Anweisungen auswirkt, müssen Sie das Paket erneut binden.

fenced_pool - Maximale Anzahl abgeschirmter Prozesse

Für **db2fmp**-Threadprozesse (Prozesse, die threadsichere gespeicherte Prozeduren und benutzerdefinierte Funktionen bedienen) stellt dieser Parameter die Anzahl Threads dar, die in jedem **db2fmp**-Prozess in den Cache gestellt werden. Für **db2fmp**-Prozesse, die keine Threadprozesse sind, stellt dieser Parameter die Anzahl der zwischengespeicherten Prozesse dar.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
-1 (**max_coordagents**), Automatic [-1; 0–64 000]

Maßeinheit
Zähler

Einschränkungen:

- Wenn dieser Parameter dynamisch aktualisiert und der Wert verringert wird, beendet der Datenbankmanager **db2fmp**-Threads oder -Prozesse nicht proaktiv, sondern stellt sie nicht mehr in den Cache, da sie zur Reduzierung der Anzahl von in den Cache gestellten db2fmp-Prozessen auf den neuen Wert verwendet werden.
- Wenn dieser Parameter dynamisch aktualisiert und der Wert erhöht wird, stellt der Datenbankmanager mehr **db2fmp**-Threads und -Prozesse in den Cache, wenn diese erstellt werden.
- Wenn dieser Parameter auf -1 gesetzt ist (Standardwert), wird der Wert des Konfigurationsparameters **max_coordagents** angenommen. Beachten Sie, dass nur der Wert von **max_coordagents** angenommen wird, nicht die automatische Einstellung oder das automatische Verhalten.
- Wenn dieser Parameter auf AUTOMATIC gesetzt ist, gilt Folgendes (dies ist auch der Standardwert):
  - Der Datenbankmanager erlaubt eine Zunahme der Anzahl der in den Cache gestellten **db2fmp**-Threads und -Prozesse auf der Basis der oberen Grenze der koordinierenden Agenten. Besonders das automatische Verhalten dieses Parameters erlaubt ihm, in Abhängigkeit von der maximalen Anzahl von koordinierenden Agenten, die der Datenbankmanager je seit dem Start (gleichzeitig) registriert hat, an Größe zuzunehmen.
  - Der diesem Parameter zugeordnete Wert stellt einen unteren Grenzwert für die Anzahl der in den Cache zu stellenden **db2fmp**-Threads und -Prozesse dar.

Empfehlung: Wenn Ihre Umgebung abgeschirmte gespeicherte Prozeduren oder benutzerdefinierte Funktionen verwendet, kann über diesen Parameter sichergestellt werden, dass zur Verarbeitung der maximalen Anzahl gleichzeitig auf der Instanz ablaufender gespeicherter Prozeduren und benutzerdefinierter Funktionen eine ausreichende Anzahl **db2fmp**-Prozesse verfügbar ist. Dadurch wird sichergestellt, dass beim Ausführen von gespeicherten Prozeduren und benutzerdefinierten Funktionen keine neuen Prozesse im abgeschirmten Modus erstellt werden müssen.

Wenn sich herausstellt, dass der Standardwert für Ihre Umgebung nicht geeignet ist, weil eine unangemessen große Menge Systemressourcen für **db2fmp**-Prozesse verwendet wird und es deshalb zu Leistungseinbußen des Datenbankmanagers kommt, kann die folgende Prozedur als Ausgangspunkt für eine Optimierung für diesen Parameter hilfreich sein:

```
fenced_pool = = Anzahl Anwendungen, die gleichzeitig Aufrufe gespeicherter
Prozeduren und UDF-Aufrufe absetzen dürfen
```

Wenn der Parameter **keepfenced** auf YES gesetzt ist, bleibt jeder im Cache-Pool erstellte **db2fmp**-Prozess bestehen und verbraucht auch dann noch Systemressourcen, wenn der Aufruf der Routine im abgeschirmten Modus verarbeitet und an den Agenten zurückgegeben wird.

Wenn **keepfenced** auf NO gesetzt ist, werden db2fmp-Prozesse, die keine Threadprozesse sind, nach Beendigung der Ausführung beendet, und es gibt keinen Cache-Pool. Multithread-**db2fmp**-Prozesse sind weiter vorhanden, jedoch werden in diesen

Prozessen keine Threads in den Pool gestellt. Dies bedeutet, dass Sie auf Ihrem System einen **db2fmp**-C-Threadprozess und einen **db2fmp**-Java-Threadprozess haben können, auch wenn **keepfenced** auf NO gesetzt ist.

In früheren Versionen hatte dieser Parameter den Namen **maxdari**.

group_plugin - Gruppen-Plug-in

Mit diesem Parameter wird der Name der Gruppen-Plug-in-Bibliothek angegeben.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
NULL [beliebige gültige Zeichenfolge]

Standardmäßig ist dieser Wert null, und DB2 verwendet die Gruppensuchfunktion (Lookup) des Betriebssystems. Das Plug-in wird für alle Gruppensuchen verwendet. Für Nichtrootinstallationen gilt, dass bei Verwendung der DB2-Bibliothek des Plug-ins für Benutzer-ID und Kennwort der Befehl **db2rfe** ausgeführt werden muss, bevor das DB2-Produkt verwendet wird.

health_mon - Überwachung mit dem Diagnosemonitor

Mit diesem Parameter können Sie angeben, ob Sie anhand verschiedener Diagnoseanzeiger eine Instanz, seine zugeordneten Datenbanken und Datenbankobjekte überwachen wollen.

Wichtig: Dieser Parameter gilt in Version 10.1 als veraltet und wird möglicherweise in einem zukünftigen Release entfernt. Der Parameter kann in Releases vor Version 10.1 weiter verwendet werden.

Eine Verwendung dieses Parameters in DB2 pureScale-Umgebungen ist nicht möglich.

Konfigurationstyp
Datenbankmanager

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
Off [On; Off]

Zugehörige Parameter

Wenn **health_mon** aktiviert ist, erfasst ein Agent Informationen zum ordnungsgemäßen Betrieb der ausgewählten Objekte. Wenn auf der Basis der von Ihnen fest-

gelegten Schwellenwerte der Betrieb eines Objekts als nicht ordnungsgemäß eingestuft wird, können automatisch Benachrichtigungen versendet und Aktionen unternommen werden. Wenn **health_mon** inaktiviert ist, wird der ordnungsgemäße Betrieb von Objekten nicht überwacht.

Sie können mit dem Befehlszeilenprozessor die Instanz und die Datenbankobjekte auswählen, die Sie überwachen wollen. Basierend auf den vom Diagnosemonitor erfassten Daten können Sie auch angeben, wann Benachrichtigungen gesendet und welche Aktionen unternommen werden sollen.

indexrec - Zeitpunkt für Indexneuerstellung

Dieser Parameter gibt an, wann der Datenbankmanager versucht, ungültige Indizes neu zu erstellen und ob eine Indexerstellung während einer aktualisierenden Recovery oder einer Wiedergabe des HADR-Protokolls in der Bereitschaftsdatenbank wiederholt wird.

Konfigurationstyp
Datenbank und Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]

UNIX Datenbankmanager
`restart` [`restart`; `restart_no_redo`; `access`; `access_no_redo`]

Windows Datenbankmanager
`restart` [`restart`; `restart_no_redo`; `access`; `access_no_redo`]

Datenbank
Systemeinstellung verwenden [`system`; `restart`; `restart_no_redo`; `access`; `access_no_redo`]

Für diesen Parameter sind fünf Einstellungen möglich:

SYSTEM
Verwenden Sie die Systemeinstellung, die in der Konfigurationsdatei des Datenbankmanagers angegeben wurde, um festzulegen, wann ungültige Indizes neu erstellt werden und ob Protokollsätze zur Indexerstellung während einer aktualisierenden Recovery oder während einer Wiedergabe des HADR-Protokolls wiederholt werden müssen.

Anmerkung: Diese Einstellung ist nur für Datenbankkonfigurationen gültig.

ACCESS
Ungültige Indizes werden erneut erstellt, wenn zum ersten Mal auf die zugrunde liegende Tabelle zugegriffen wird. Alle vollständig protokollierten Indexerstellungen werden während einer aktualisierenden Recovery oder während einer Wiedergabe des HADR-Protokolls wiederholt. Wenn HADR gestartet wird und eine HADR-Übernahme stattfindet, werden alle ungülti-

gen Indizes nach der Übernahme, wenn zum ersten Mal auf die zugrunde liegende Tabelle zugegriffen wird, neu erstellt.

ACCESS_NO_REDO
Ungültige Indizes werden erneut erstellt, wenn zum ersten Mal auf die zugrunde liegende Tabelle zugegriffen wird. Keine der vollständig protokollierten Indexerstellungen wird während einer aktualisierenden Recovery oder während einer Wiedergabe des HADR-Protokolls wiederholt; diese Indizes bleiben ungültig. Wenn HADR gestartet wird und eine HADR-Übernahme stattfindet, werden alle ungültigen Indizes nach der Übernahme, wenn zum ersten Mal auf die zugrunde liegende Tabelle zugegriffen wird, neu erstellt. Der Zugriff auf die zugrunde liegenden Tabellen in der neuen Primärdatenbank bewirkt einen Indexrebuild. Dies führt dazu, dass Protokollsätze geschrieben und an die neue Bereitschaftsdatenbank gesendet werden. Dies wiederum führt dazu, dass die Indizes in der Bereitschaftsdatenbank ungültig gemacht werden.

RESTART
Die Standardeinstellung für **indexrec**. Ungültige Indizes werden neu erstellt, wenn der Befehl **RESTART DATABASE** explizit oder implizit abgesetzt wird. Alle vollständig protokollierten Indexerstellungen werden während einer aktualisierenden Recovery oder während einer Wiedergabe des HADR-Protokolls wiederholt. Wenn HADR gestartet wird und eine HADR-Übernahme stattfindet, werden alle ungültigen Indizes nach der Übernahme neu erstellt.

Anmerkung: In einer DB2 pureScale-Umgebung werden Indizes nur bei einer Recovery nach Absturz einer Gruppe erneut erstellt, nicht im Rahmen der Recovery nach Absturz eines Members.

Anmerkung: Wenn beim abnormalen Beenden einer Datenbank, die mit Anwendungen verbunden ist, der Parameter `autorestart` aktiviert wird, wird beim Herstellen einer Verbindung von einer Anwendung zu einer Datenbank implizit der Befehl **RESTART DATABASE** abgesetzt. Wenn der Befehl nicht abgesetzt wird, werden die ungültigen Indizes erneut erstellt, sobald der nächste Zugriff auf die zugrunde liegende Tabelle erfolgt.

RESTART_NO_REDO
Ungültige Indizes werden neu erstellt, wenn der Befehl **RESTART DATABASE** explizit oder implizit abgesetzt wird. Keine der vollständig protokollierten Indexerstellungen wird während einer aktualisierenden Recovery oder während einer Wiedergabe des HADR-Protokolls wiederholt; stattdessen werden diese Indizes neu erstellt, wenn die aktualisierende Recovery beendet wird oder wenn die HADR-Übernahme stattfindet. Die Übernahme bewirkt einen Indexrebuild für die zugrunde liegenden Tabellen in der Primärdatenbank. Dies führt dazu, dass Protokollsätze geschrieben und an die neue Bereitschaftsdatenbank gesendet werden. Dies wiederum führt dazu, dass die Indizes in der Bereitschaftsdatenbank ungültig gemacht werden.

Wenn beim abnormalen Beenden einer Datenbank, die mit Anwendungen verbunden ist, der Parameter `autorestart` aktiviert wird, wird beim Herstellen einer Verbindung von einer Anwendung zu einer Datenbank implizit der Befehl **RESTART DATABASE** abgesetzt. Wenn der Befehl nicht abgesetzt wird, werden die ungültigen Indizes erneut erstellt, sobald der nächste Zugriff auf die zugrunde liegende Tabelle erfolgt.

Indizes können ungültig werden, wenn nicht behebbare Plattenfehler auftreten. Wenn dabei die Daten selbst beschädigt werden, können die Daten verloren gehen.

Wird jedoch ein Index beschädigt, kann der Index durch Neuerstellung wiederhergestellt werden. Wird ein Index neu erstellt, während Benutzer mit der Datenbank verbunden sind, können zwei Probleme auftreten:

- Während der Erstellung der Indexdatei könnte es zu einer unerwarteten Verschlechterung der Antwortzeit kommen. Benutzer, die auf die Tabelle zugreifen und diesen bestimmten Index verwenden, müssen auf die Neuerstellung des Index warten.
- Nach der Indexneuerstellung könnten unerwartete, aktive Sperren beibehalten werden, insbesondere wenn die Benutzertransaktion, die die Indexneuerstellung ausgelöst hat, keine COMMIT- oder ROLLBACK-Operation ausgeführt hat.

Empfehlung: Die beste Option für diesen Parameter auf einem Server mit vielen Benutzern, wenn die Zeit für den Neustart keine kritische Rolle spielt, ist die Indexneuerstellung beim Neustart der Datenbank (**DATABASE RESTART**) als Teil eines Vorgangs, mit dem die Datenbank nach einem Systemabsturz wiederhergestellt und online verfügbar gemacht wird.

Wenn dieser Parameter auf ACCESS oder auf ACCESS_NO_REDO gesetzt wird, verschlechtert sich während der Indexneuerstellung die Leistung des Datenbankmanagers. Jeder Benutzer, der auf den Index oder die Tabelle zugreift, der bzw. die gerade neu erstellt wird, muss zunächst auf die Beendigung der Indexneuerstellung warten.

Wenn dieser Parameter auf RESTART gesetzt ist, dauert der Neustart der Datenbank wegen der Indexneuerstellung länger, jedoch wird die normale Verarbeitung nicht mehr beeinträchtigt, sobald die Datenbank wieder online verfügbar ist.

Anmerkung: Bei der Datenbankrecovery werden alle ausführbaren Dateien von SQL-Prozeduren im Dateisystem entfernt, die zu der gerade wiederhergestellten Datenbank gehören. Wenn der Parameter **indexrec** auf RESTART gesetzt ist, werden alle ausführbaren Dateien von SQL-Prozeduren aus dem Datenbankkatalog extrahiert und in das Dateisystem zurückgestellt, wenn die nächste Verbindung zur Datenbank hergestellt wird. Wenn der Parameter **indexrec** nicht auf RESTART gesetzt ist, wird die ausführbare Datei einer SQL-Prozedur nur bei der ersten Ausführung der SQL-Prozedur in das Dateisystem extrahiert.

Der Unterschied zwischen den Werten RESTART und RESTART_NO_REDO bzw. zwischen den Werten ACCESS und ACCESS_NO_REDO ist nur von Bedeutung, wenn die vollständige Protokollierung für Indexerstellungsoperationen (z. B. **CREATE INDEX** und **REORG INDEX**) oder für eine Indexneuerstellung aktiviert ist. Sie können die Protokollierung aktivieren, indem Sie den Datenbankkonfigurationsparameter **logindexbuild** oder LOG INDEX BUILD beim Ändern einer Tabelle aktivieren. Wenn Sie **indexrec** entweder auf RESTART oder auf ACCESS setzen, kann für Operationen, die eine protokollierte Indexerstellung einbeziehen, eine aktualisierende Recovery durchgeführt werden, ohne dass das Indexobjekt einen ungültigen Status aufweist; dies hätte zur Folge, dass der Index später neu erstellt werden müsste.

instance_memory - Instanzspeicher

Dieser Parameter definiert die maximale Speichermenge, die der Datenbankpartition zugeordnet werden kann, wenn Sie DB2-Datenbankprodukte mit Speichernutzungsbegrenzungen einsetzen oder diesen Parameter auf einen bestimmten Wert setzen. Ansonsten ermöglicht die Einstellung AUTOMATIC ein bedarfsgerechtes Anwachsen der Instanzspeichermenge.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Standardwert [Bereich]

32-Bit-Plattformen
Automatic [0 - 1 000 000]

64-Bit-Plattformen
Automatic [0 - 68 719 476 736]

DB2 Express Edition und DB2 Express-C
Automatic [0 - 1 048 576]

DB2 Workgroup Server Edition
Automatic [0 - 4 194 304]

Maßeinheit
Seiten (4 KB)

Zuordnung
Nicht zutreffend

Freigabe
Nicht zutreffend

Der Standardwert von **instance_memory** ist AUTOMATIC. Die Einstellung AUTOMATIC bewirkt, dass der verwendete Wert bei der Aktivierung der Datenbankpartition berechnet wird. Der berechnete Wert liegt liegt zwischen 75 und 95 Prozent des physischen Arbeitsspeichers (RAM) des Systems, wobei der Prozentsatz umso höher liegt, je größer das System ist. Für DB2-Datenbankprodukte mit Speichernutzungsbegrenzungen wird der berechnete Wert durch das durch die Produktlizenz zugelassene Maximum begrenzt. Bei Datenbankpartitionsservern mit mehreren logischen Datenbankpartitionen wird dieser berechnete Wert durch die Anzahl der logischen Datenbankpartitionen dividiert.

Ab Version 9.7 Fixpack 1 und Version 9.5 Fixpack 5 legt der berechnete Wert für die Einstellung AUTOMATIC keine Begrenzung für den Speicher fest, der in der gesamten Instanz für DB2-Datenbankprodukte ohne Speichernutzungsbegrenzungen zugeordnet wird. Bei Version 9.7 und Version 9.5 mit Fixpack 4 oder früheren Versionen stellt der berechnete Wert für die Einstellung AUTOMATIC eine Begrenzung für alle DB2-Datenbankprodukte dar.

Dynamisches Aktualisieren des Parameters instance_memory

- Für dynamische Aktualisierungen am Parameter **instance_memory** ist eine Instanzzuordnung (ATTACH) erforderlich. Weitere Informationen finden Sie unter dem Befehl **ATTACH**.
- Für DB2-Datenbankprodukte mit Speichernutzungsbegrenzungen müssen dynamische Aktualisierungen am Parameter **instance_memory** einen Wert, der kleiner als die Lizenzbegrenzung ist, oder den Wert AUTOMATIC angeben. Andernfalls schlägt die Aktualisierung fehl und die Fehlernachricht SQL5130N wird zurückgegeben.

- Dynamische Aktualisierungen am Parameter **instance_memory** müssen einen Wert, der kleiner als die Kapazität des physischen Arbeitsspeichers (RAM) ist, oder den Wert AUTOMATIC angeben. Andernfalls wird die Aktualisierung aufgeschoben, bis der nächste Befehl **db2start** ausgeführt wird, und die Warnung SQL1362W zurückgegeben.
- Dynamische Aktualisierungen am Parameter **instance_memory** müssen einen Wert angeben, der größer als die zurzeit genutzte Menge an Instanzspeicher ist. Andernfalls wird die Aktualisierung aufgeschoben, bis die Instanz erneut gestartet wird, und die Warnung SQL1362W zurückgegeben. Die zurzeit verwendete Menge an Instanzspeicher kann durch Subtrahieren des Werts im Feld für Cachespeicher (*Cached memory*) vom Wert für aktuelle Nutzung (*Current usage*) in der Ausgabe des Befehls **db2pd -dbptnmem** bestimmt werden. Der Mindestwert ist dabei der höchste Wert für die zurzeit verwendete Instanzspeichermenge unter allen Datenbankpartitionen.
- Wenn der Parameter **instance_memory** auf einen höheren Wert als die Kapazität an physischem Speicher (RAM) gesetzt wird, schlägt der nächste Befehl **db2start** fehl, den Sie ausführen, und die Fehlernachricht SQL1220N wird zurückgegeben.
- Wenn der Parameter **instance_memory** dynamisch auf den Wert AUTOMATIC aktualisiert wird, wird der tatsächliche Wert unverzüglich neu berechnet.

Einschränkung für Instanzen in Umgebungen mit partitionierten Datenbanken
Sie sollten in Umgebungen mit partitionierten Datenbanken keinen bestimmten Wert für den Parameter **instance_memory** verwenden. Die Verwendung eines bestimmten Werts für den Parameter **instance_memory** wird in Umgebungen mit partitionierten Datenbanken nicht empfohlen, weil **instance_memory** ein Konfigurationsparameter des Datenbankmanagers ist, für den keine Angabe verschiedener Werte für verschiedene Datenbankpartitionen möglich ist. Deshalb ist es schwierig, eine Einstellung, die sich für alle Datenbankpartitionen eignet, festzulegen, da die Datenbankpartitionen verschiedene Speicheranforderungen haben können.

Steuern der DB2-Speicherbelegung:

Die Speicherbelegung von DB2 variiert je nach Auslastung und Konfiguration. Darüber hinaus stellt die automatische Leistungsoptimierung des Parameters **database_memory** einen Faktor dar, wenn sie aktiviert ist. Die automatische Leistungsoptimierung des Parameters **database_memory** ist aktiviert, wenn der Parameter **database_memory** auf den Wert AUTOMATIC gesetzt ist und die automatische Speicheroptimierungsfunktion (STMM, Self-Tuning Memory Manager) aktiv ist.

Wenn die Instanz auf einem DB2-Datenbankprodukt ohne Speichernutzungsbegrenzungen ausgeführt wird und der Parameter **instance_memory** auf AUTOMATIC gesetzt ist, wird keine Begrenzung für **instance_memory** festgelegt. Der Datenbankmanager ordnet Systemspeicher nach Bedarf zu. Wenn die automatische Leistungsoptimierung des Parameters **database_memory** aktiviert ist, aktualisiert STMM die Konfiguration, um eine optimale Leistung zu erreichen, wobei der verfügbare Systemspeicher überwacht wird. Diese Überwachung des verfügbaren Speichers stellt sicher, dass nicht zu viel Systemspeicher reserviert wird.

Wenn die Instanz auf einem DB2-Datenbankprodukt mit Speichernutzungsbegrenzungen ausgeführt wird oder der Parameter **instance_memory** auf einen bestimmten Wert gesetzt wird, wird eine Begrenzung für **instance_memory** festgelegt. Der Datenbankmanager ordnet Systemspeicher

bis zu dieser Begrenzung zu. Die Anwendung kann Speicherzuordnungsfehler empfangen, wenn diese Begrenzung erreicht wird. Darüber hinaus sind die folgenden Punkte zu beachten:

- Wenn die automatische Leistungsoptimierung des Parameters **database_memory** aktiviert ist und der Parameter **instance_memory** auf einen bestimmten Wert gesetzt wird, aktualisiert STMM die Konfiguration, um eine optimale Leistung zu erreichen, wobei genügend Instanzspeicher freigehalten wird. Dies stellt sicher, dass genügend Instanzspeicher verfügbar ist, um flüchtige Speicheranforderungen zu bedienen. Der Systemspeicher wird nicht überwacht.
- Wenn die automatische Leistungsoptimierung des Parameters **database_memory** aktiviert ist und der Parameter **instance_memory** auf den Wert AUTOMATIC gesetzt wird, was den Fall darstellt, in dem eine Begrenzung für den Parameter **instance_memory** in einem DB2-Datenbankprodukt mit Speichernutzungsbegrenzung festgelegt wird, aktualisiert STMM die Konfiguration, um eine optimale Leistung zu erreichen, wobei der verfügbare Systemspeicher überwacht und genügend Instanzspeicher freigehalten wird.

Überwachen der Instanzspeichernutzung

Mithilfe des Befehls **db2pd -dbptnmem** können Sie Details zur Nutzung des Instanzspeichers anzeigen.

Mit der neuen Tabellenfunktion ADMIN_GET_MEM_USAGE können Sie die Gesamtnutzung des Instanzspeichers durch eine DB2-Instanz für eine bestimmte Datenbankpartition oder für alle Datenbankpartitionen ermitteln. Diese Tabellenfunktion gibt außerdem den aktuellen oberen Grenzwert zurück.

Wenn der gemeinsam genutzte FCM-Speicher (FCM = Fast Communication Manager) zugeordnet wird, wird der Anteil der einzelnen lokalen Datenbankpartitionen an der gesamten Größe des gemeinsam genutzten FCM-Speichers für das System in der durch den Parameter **instance_memory** der betreffenden Datenbankpartition angegebenen Nutzung berücksichtigt.

intra_parallel - Partitionsinterne Parallelität aktivieren

Dieser Parameter gibt an, ob der Datenbankmanager partitionsinterne Parallelität verwenden kann.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
NO (0) [SYSTEM (-1), NO (0), YES (1)]

Der Wert -1 bewirkt, dass der Parameter auf YES oder NO gesetzt wird; abhängig von der Hardware, auf welcher der Datenbankmanager ausgeführt wird.

Anmerkung: Der Standardwert unterliegt nach der ursprünglichen Datenbankerstellung Änderungen durch den DB2 Configuration Advisor.

Anmerkung:

- Bei der parallelen Indexerstellung wird dieser Konfigurationsparameter nicht verwendet.
- Wenn Sie diesen Parameterwert ändern, werden Pakete möglicherweise erneut an die Datenbank gebunden und es kann zu Leistungseinbußen kommen.
- Die Einstellung für **intra_parallel** kann in einer Anwendung durch einen Aufruf der Prozedur ADMIN_SET_INTRA_PARALLEL überschrieben werden. Sowohl die Einstellung für **intra_parallel** als auch der in einer Anwendung durch die Prozedur ADMIN_SET_INTRA_PARALLEL festgelegte Wert kann in einer Workload durch Festlegen des Attributs MAXIMUM DEGREE in einer Workloaddefinition überschrieben werden.

java_heap_sz - Maximale Zwischenspeichergröße für Java-Interpreter

Dieser Parameter legt die maximale Größe des Zwischenspeichers fest, der von dem zur Verarbeitung von gespeicherten Java- DB2-Prozeduren und benutzerdefinierten Funktionen gestarteten Java-Interpreter verwendet wird.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]

HP-UX
4096 [0 - 524 288]

Alle anderen Betriebssysteme
2048 [0 - 524 288]

Maßeinheit
Seiten (4 KB)

Zuordnung
Wenn eine gespeicherte Java-Prozedur oder eine benutzerdefinierte Funktion gestartet wird

Freigabe
Wenn der db2fmp-Prozess (abgeschirmt) oder der db2agent-Prozess (gesichert) beendet wird

Es gibt einen Zwischenspeicher für jeden db2fmp-Prozess, der eine gespeicherte Java-Prozedur ausführt. Für Multithread-db2fmp-Prozesse werden mehrere Anwendungen, die threadsichere abgeschirmte Routinen verwenden, von einem einzigen Zwischenspeicher bedient. In allen Fällen ordnen nur die Prozesse, die in Java geschriebene benutzerdefinierte Funktionen oder gespeicherte Prozeduren ausfüh-

ren, diesen Speicher zu. Bei partitionierten Datenbanksystemen wird in jeder Datenbankpartition derselbe Wert verwendet.

XML-Daten werden umgesetzt, wenn sie als Parameter IN, OUT oder INOUT an gespeicherte Prozeduren übergeben werden. Wenn Sie mit gespeicherten Java-Prozeduren arbeiten, müssen eventuell die Zwischenspeichergröße je nach Anzahl und Größe der XML-Argumente sowie die Anzahl externer gespeicherter Prozeduren, die gleichzeitig ausgeführt werden, erhöht werden.

jdk_path - Installationspfad für Software Developer's Kit für Java ()

Dieser Parameter gibt das Verzeichnis an, in dem das Software Developer's Kit (SDK) für Java installiert ist, das verwendet wird, um gespeicherte Java-Prozeduren und benutzerdefinierte Funktionen auszuführen. **CLASSPATH** und andere vom Java-Interpreter verwendete Umgebungsvariablen werden aus dem Wert dieses Parameters errechnet.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
NULL [Gültiger Pfad]

Wenn das SDK für Java zusammen mit Ihrem DB2-Produkt installiert wurde, ist dieser Parameter richtig definiert. Wenn Sie allerdings den Datenbankmanagerparameter (dbm cfg) zurücksetzen, müssen Sie angeben, wo das SDK für Java installiert ist.

keepfenced - Abgeschirmten Prozess beibehalten

Dieser Parameter gibt an, ob ein Prozess im abgeschirmten Modus beibehalten wird, nachdem der Aufruf einer Routine im abgeschirmten Modus beendet ist. Prozesse im abgeschirmten Modus werden als getrennte Systementitäten erstellt, um vom Benutzer geschriebenen Code im abgeschirmten Modus vom Agentenprozess des Datenbankmanagers zu isolieren. Dieser Parameter gilt nur für Datenbankserver.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

s

Parametertyp
Konfigurierbar

Standardwert [Bereich]
Yes [Yes; No]

Wenn **keepfenced** auf No gesetzt ist und die ausgeführte Routine nicht threadsicher ist, wird für jeden Aufruf im abgeschirmten Modus ein neuer Prozess im abgeschirmten Modus erstellt und zerstört. Wenn **keepfenced** auf no gesetzt ist und die ausgeführte Routine threadsicher ist, wird der Prozess im abgeschirmten Modus beibehalten, der für den Aufruf erstellte Thread wird jedoch beendet. Wenn **keepfenced** auf yes gesetzt ist, wird ein Thread oder Prozess im abgeschirmten Modus für nachfolgende Aufrufe im abgeschirmten Modus wiederverwendet. Wird der Datenbankmanager gestoppt, werden alle noch anstehenden Threads und Prozesse im abgeschirmten Modus beendet.

Wenn dieser Parameter auf yes gesetzt wird, werden vom Datenbankmanager zusätzliche Systemressourcen für jeden Prozess im abgeschirmten Modus in Anspruch genommen, der aktiviert wird, solange die durch den Parameter **fenced_pool** definierte Anzahl noch nicht erreicht ist. Ein neuer Prozess wird nur dann erstellt, wenn zur Verarbeitung eines nachfolgenden Aufrufs einer Routine im abgeschirmten Modus kein Prozess im abgeschirmten Modus verfügbar ist. Dieser Parameter wird ignoriert, wenn **fenced_pool** auf 0 gesetzt ist.

Empfehlung: In einer Umgebung, in der die Anzahl der Anforderungen im abgeschirmten Modus im Vergleich zu den übrigen Anforderungen groß ist und Systemressourcen nicht begrenzt sind, kann dieser Parameter auf yes gesetzt werden. Dadurch wird die Leistung des Prozesses im abgeschirmten Modus erhöht, weil die zusätzliche Verarbeitungszeit zur Ersterstellung eines Prozesses im abgeschirmten Modus vermieden wird, da ein bereits vorhandener Prozess im abgeschirmten Modus zur Verarbeitung des Aufrufs verwendet wird. Dies erspart vor allem bei Java-Routinen den Aufwand, die JVM (Java Virtual Machine) zu starten, was eine erhebliche Leistungssteigerung bedeutet.

Zum Beispiel könnte bei einer OLTP-Bankanwendung (OLTP - Online-Transaktionsprogramm) für Soll- und Haben-Transaktionen der Code, mit dem jede Transaktion ausgeführt wird, in einer gespeicherten Prozedur ausgeführt werden, die in einem Prozess im abgeschirmten Modus ausgeführt wird. In dieser Anwendung wird die Hauptauslastung aus Prozessen im abgeschirmten Modus heraus ausgeführt. Wenn dieser Parameter auf no gesetzt wird, ist für jede Transaktion die zusätzliche Verarbeitungszeit zum Erstellen eines neuen Prozesses im abgeschirmten Modus erforderlich, wodurch die Leistung erheblich beeinträchtigt wird. Wenn dieser Parameter jedoch auf yes gesetzt wird, versucht jede Transaktion, einen vorhandenen Prozess im abgeschirmten Modus zu verwenden, wodurch die zusätzliche Verarbeitungszeit entfällt.

In früheren Versionen von DB2 hatte dieser Parameter den Namen **keepdari**.

local_gssplugin - GSS-API-Plug-in für lokale Berechtigung auf Instanzebene

Dieser Parameter gibt den Namen der GSS-API-Plug-in-Standardbibliothek an, die zur lokalen Berechtigung auf Instanzebene verwendet wird, wenn der Konfigurationsparameter **authentication** des Datenbankmanagers auf den Wert GSSPLUGIN oder GSS_SERVER_ENCRYPT gesetzt ist.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
NULL [beliebige gültige Zeichenfolge]

max_connections - Maximale Anzahl von Clientverbindungen

Dieser Parameter gibt die maximal zulässige Anzahl Clientverbindungen pro Member an.

Konfigurationstyp
Datenbankmanager

Parametertyp
Online konfigurierbar

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Datenbankserver oder Connect-Server mit lokalen und fernen Clients (für **max_connections**, **max_coordagents**, **num_initagents**, **num_poolagents** und auch für **federated_async**, wenn Sie eine Umgebung mit föderierten Datenbanken verwenden)

Standardwert [Bereich]
-1 und AUTOMATIC (**max_coordagents**) [-1 und AUTOMATIC, 1 - 64000]

Die Einstellung -1 bedeutet, dass der dem Parameter **max_coordagents** zugeordnete Wert und nicht die automatische Einstellung bzw. das automatische Verhalten verwendet wird. Die Einstellung AUTOMATIC bedeutet, dass der Datenbankmanager den Wert für diesen Parameter mit der am besten geeigneten Technik auswählt. Die Einstellung AUTOMATIC ist ein Aktivierungs- bzw. Inaktivierungsschalter in der Konfigurationsdatei und vom entsprechenden Wert unabhängig. Daher können beide Einstellungen (-1 und AUTOMATIC) die Standardeinstellung sein.

Detaillierte Informationen finden Sie in „Einschränkungen und Verhalten bei der Konfiguration von 'max_coordagents' und 'max_connections'" auf Seite 748.

Der Konzentrator

Der Konzentrator ist inaktiviert, wenn **max_connections** kleiner oder gleich **max_coordagents** ist. Der Konzentrator ist aktiviert, wenn **max_connections** größer als **max_coordagents** ist.

Dieser Parameter steuert die maximale Anzahl der Clientanwendungen, die mit einem Member der Instanz verbunden sein können. In der Regel wird jede Anwendung einem Koordinatoragenten zugeordnet. Der Agent vereinfacht die Operationen zwischen der Anwendung und der Datenbank. Die Konzentratorfunktion wird nicht aktiviert, wenn der Standardwert für diesen Parameter verwendet wird. Daher arbeitet jeder Agent in seinem eigenen privaten Speicher und verwendet Ressourcen des Datenbankmanagers und globale Datenbankressourcen wie z. B. den

Pufferpool gemeinsam mit anderen Agenten. Die Konzentratorfunktion wird hingegen aktiviert, wenn der Parameter auf einen Wert gesetzt wird, der den Standardwert überschreitet.

Verwendung

Die maximale Anzahl von Clientverbindungen kann größer als der Wert sein, der durch den Konfigurationsparameter **max_connections** festgelegt ist, wenn der Benutzer eine der folgenden Berechtigungen hat:

- SYSADM
- SYSCTRL
- SYSMAIN

Anmerkung: Das Umgehen der maximalen Anzahl von Clientverbindungen, die durch den Konfigurationsparameter **max_connections** festgelegt ist, kann aus verschiedenen Gründen sinnvoll sein, wie z. B. bei folgenden Aufgaben:

- Verwaltungstasks
- Erfassen von Momentaufnahmen
- Kritische Fehlerbehebungstasks
- Wartungstasks (z. B. Erzwingen der Trennung bestimmter Benutzer vom System)

max_connretries - Anzahl Wiederholungen für Verbindungsaufbau zum Knoten

Dieser Parameter gibt an, wie häufig höchstens versucht wird, eine Netzverbindung zwischen zwei DB2-Membern aufzubauen.

Konfigurationstyp
Datenbankmanager

Gilt für
Partitionierten Datenbankserver mit lokalen und fernen Clients

DB2 pureScale-Server

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
5 [0–100]

Wenn der Verbindungsaufbau zwischen zwei DB2-Membern fehlschlägt (z. B. bei Erreichen des im Parameter **conn_elapse** angegebenen Werts), gibt **max_connretries** an, wie viele Wiederholungen durchgeführt werden dürfen, um die Verbindung zu einem DB2-Member herzustellen. Bei Überschreitung des für diesen Parameter angegebenen Werts wird eine Fehlernachricht zurückgegeben.

max_coordagents - Maximale Anzahl koordinierender Agenten

Dieser Parameter wird verwendet, um die Anzahl koordinierender Agenten zu begrenzen.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
`200, Automatic [-1; 0–64 000]`

Die Einstellung `-1` wird in den Wert `200` umgesetzt.

Detaillierte Informationen finden Sie in „Einschränkungen und Verhalten bei der Konfiguration von 'max_coordagents' und 'max_connections'" auf Seite 748.

Der Konzentrator

Wenn der Konzentrator nicht aktiviert ist, d. h., wenn **max_connections** kleinergleich **max_coordagents** ist, legt dieser Parameter die maximale Anzahl koordinierender Agenten fest, die gleichzeitig auf einem Serverknoten vorhanden sein können.

Für jede lokale oder ferne Anwendung, die eine Verbindung zu einer Datenbank oder einer Instanz herstellt, wird ein koordinierender Agent aufgerufen. Anforderungen, für die eine Instanzverbindung erforderlich ist, sind z. B. **CREATE DATABASE**, **DROP DATABASE** und Befehle des Datenbanksystemmonitors.

Wenn der Konzentrator aktiviert ist, d. h., wenn **max_connections** größer als **max_coordagents** ist, kann es mehr Verbindungen als Koordinatoragenten für die Verbindungen geben. Eine Anwendung ist nur dann aktiv, wenn sie von einem Koordinatoragenten bedient wird. Andernfalls ist die Anwendung inaktiv. Anforderungen von einer aktiven Anwendung werden vom Koordinatoragenten der Datenbank (und in SMP- oder MPP-Konfigurationen von Subagenten) bedient. Anforderungen von einer inaktiven Anwendung werden in eine Warteschlange gestellt, bis ein Datenbankkoordinatoragent zur Bedienung der Anwendung zugeordnet wird, wenn diese aktiv wird. Daher kann dieser Parameter zur Steuerung der Systembelastung verwendet werden.

Verwendung

Die maximale Anzahl von koordinierenden Agenten kann größer als der Wert sein, der durch den Konfigurationsparameter **max_coordagents** festgelegt ist, wenn der Benutzer eine der folgenden Berechtigungen hat:

- SYSADM
- SYSCTRL
- SYSMAIN

Anmerkung: Das Umgehen der maximalen Anzahl von koordinierenden Agenten, die durch den Konfigurationsparameter **max_coordagents** festgelegt ist, kann aus verschiedenen Gründen sinnvoll sein, wie z. B. bei folgenden Aufgaben:

- Verwaltungstasks
- Erfassen von Momentaufnahmen
- Kritische Fehlerbehebungstasks
- Wartungstasks (z. B. Erzwingen der Trennung bestimmter Benutzer vom System)

max_querydegree - max_querydegree - Maximaler Grad der Parallelität bei Abfragen

Dieser Parameter gibt den maximalen Grad partitionsinterner Parallelität an, der für SQL-Anweisungen verwendet wird, die auf dieser Instanz des Datenbankmanagers ausgeführt werden. Eine SQL-Anweisung verwendet bei ihrer Ausführung innerhalb einer Datenbankpartition nicht mehr als diese Anzahl paralleler Operationen.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Anweisungsgrenzwert

Standardwert [Bereich]
`-1 (ANY) [ANY, 1 - 32 767] (ANY bedeutet 'vom System festgelegt')`

Anmerkung: Der Standardwert unterliegt nach der ursprünglichen Datenbankerstellung Änderungen durch den DB2 Configuration Advisor.

Der Konfigurationsparameter **intra_parallel** muss aktiviert (`YES`) sein, damit die Datenbankpartition die partitionsinterne Parallelität für SQL-Anweisungen verwenden kann. Der Parameter **intra_parallel** ist für die parallele Indexerstellung nicht mehr erforderlich.

Der Standardwert für diesen Konfigurationsparameter lautet `-1`. Dieser Wert bedeutet, dass das System den vom Optimierungsprogramm festgelegten Parallelitätsgrad verwendet. Andernfalls wird der vom Benutzer angegebene Wert verwendet.

Anmerkung: Der Grad der Parallelität für eine SQL-Anweisung kann bei der Kompilierung der Anweisung mithilfe des Sonderregisters CURRENT DEGREE oder der Bindeoption **DEGREE** angegeben werden.

Der maximale Grad der Parallelität für eine aktive Anwendung bei Abfragen kann mit dem Befehl **SET RUNTIME DEGREE** geändert werden. Der zur Laufzeit tatsächlich verwendete Parallelitätsgrad ist der niedrigste der folgenden Werte:

- Konfigurationsparameter **max_querydegree**
- Parallelitätsgrad der Anwendung zur Laufzeit
- Parallelitätsgrad bei der Kompilierung der SQL-Anweisung

Dieser Konfigurationsparameter gilt nur für Abfragen.

max_time_diff - Maximale Zeitdifferenz zwischen Membern ()

Dieser Parameter gibt die maximale Zeitdifferenz an, die in einer DB2 pureScale-Umgebung zwischen den in der Knotenkonfigurationsdatei aufgelisteten Membern zulässig ist.

Konfigurationstyp
Datenbankmanager

Gilt für
Member mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]

In DB2 pureScale-Umgebungen
1 [1 - 1 440]

Außerhalb von DB2 pureScale-Umgebungen
60 [1 - 1 440]

Maßeinheit
Minuten

Jedes Member verfügt über eine eigene Systemuhr. Der Zeitunterschied zwischen den Systemuhren von zwei oder mehr Membern wird regelmäßig überprüft. Wenn die Zeitdifferenz zwischen den Systemuhren den Wert überschreitet, der im Parameter **max_time_diff** angegeben ist, werden in den db2diag-Protokolldateien Warnungen protokolliert.

Um in einer DB2 pureScale-Umgebung sicherzustellen, dass die Member synchron zueinander bleiben, ist eine NTP-Konfiguration (NTP - Network Time Protocol) erforderlich, die regelmäßig auf den einzelnen Membern überprüft wird. Wenn der NTP-Dämon nicht gefunden werden kann, werden entsprechende Warnungen in den db2diag-Protokolldateien protokolliert.

Der Fehler SQL1473N wird in partitionierten Datenbankumgebungen gemeldet, in denen die Systemuhr mit der virtuellen Zeitmarke (Virtual Time Stamp, VTS) verglichen wird, die in der Protokollsteuerdatei SQLOGCTL.LFH gespeichert ist. Wenn die Zeitmarke in der Protokollsteuerdatei .LFH kleiner als die Systemzeit ist, wird die Zeit im Datenbankprotokoll auf die virtuelle Zeitmarke gesetzt, bis die Systemuhr mit dieser Zeitmarke übereinstimmt.

Der DB2-Datenbankmanager verwendet die Weltzeit (Coordinated Universal Time, UTC), damit unterschiedliche Zeitzonen beim Festlegen des Parameters **max_time_diff** keine Rolle spielen. Die Weltzeit (UTC) ist identisch mit der Westeuropäischen Zeit (Greenwich Mean Time).

maxagents - Maximale Anzahl von Agenten

Dieser Parameter ist seit Version 9.5 veraltet, wird jedoch von Datenservern und Clients mit Versionen vor Version 9.5 weiterhin verwendet. Jeder für diesen Konfigurationsparameter angegebene Wert wird vom Datenbankmanager in DB2 Version 9.5 oder neueren Releases ignoriert.

Anmerkung: Die nachfolgenden Informationen gelten nur für Datenserver und Clients einer Version vor Version 9.5.

Mit diesem Parameter wird die maximale Anzahl von Datenbankmanageragenten (Koordinatoragenten und Subagenten) angegeben, die zu einem gegebenen Zeitpunkt zur Verfügung stehen, um Anwendungsanforderungen zu empfangen.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
`200` [`1 - 64 000`]

`400` [`1 - 64 000`] auf partitioniertem Datenbankserver mit lokalen und fernen Clients

Maßeinheit
Zähler

Wenn Sie die Anzahl koordinierender Agenten begrenzen möchten, verwenden Sie den Parameter **max_coordagents**.

Dieser Parameter kann in Umgebungen mit eingeschränkten Speicherkapazitäten nützlich sein, um den Gesamtspeicherbedarf des Datenbankmanagers zu begrenzen, da jeder zusätzliche Agent zusätzlichen Speicher benötigt.

Empfehlung: Der Wert des Parameters **maxagents** sollte mindestens der Summe der Werte für **maxappls** aller Datenbanken, auf die gleichzeitig zugegriffen werden kann, entsprechen. Wenn die Anzahl der Datenbanken größer als der Wert des Parameters **numdb** ist, dann besteht die sicherste Methode zur Angabe dieses Werts darin, das Produkt von **numdb** und dem größten Wert für **maxappls** zu bilden.

Für jeden weiteren Agenten sind zusätzliche Verarbeitungsressourcen erforderlich, die beim Starten des Datenbankmanagers zugeordnet werden.

Falls bei dem Versuch, eine Verbindung zu einer Datenbank herzustellen, Speicherfehler vorkommen, führen Sie die folgenden Konfigurationsanpassungen aus:

- Erhöhen Sie in einer Umgebung mit nicht partitionierten Datenbanken, in der die abfrageinterne Parallelität nicht aktiviert ist, den Wert des Datenbankkonfigurationsparameters **maxagents**.
- Erhöhen Sie in einer Umgebung mit partitionierten Datenbanken oder einer Umgebung, in der abfrageinterne Parallelitäten aktiviert sind, den höheren der beiden Parameter **maxagents** oder **max_coordagents**.

maxcagents - Maximale Anzahl gleichzeitig aktiver Agenten

Dieser Parameter ist in Version 9.5 veraltet, wird jedoch von Datenservern und Clients mit Versionen vor Version 9.5 weiterhin verwendet. Jeder für diesen Konfigurationsparameter angegebene Wert wird vom Datenbankmanager in DB2 Version 9.5 ignoriert.

Anmerkung: Die nachfolgenden Informationen gelten nur für Datenserver und Clients einer Version vor Version 9.5.

Dieser Parameter wird zur Steuerung der Systembelastung in Phasen hoher gleichzeitiger Anwendungsaktivität verwendet, indem die maximale Anzahl von Datenbankmanageragenten festgelegt wird, die gleichzeitig eine Datenbankmanagertransaktion ausführen können.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
-1 (**max_coordagents**) [-1; 1 – **max_coordagents**]

Maßeinheit
Zähler

Dieser Parameter schränkt nicht die Anzahl der Anwendungen ein, die mit einer Datenbank verbunden sein können. Er begrenzt nur die Anzahl der Datenbankmanageragenten, die gleichzeitig vom Datenbankmanager verarbeitet werden können, wodurch der Bedarf an Systemressourcen in Phasen sehr starker Belastung eingeschränkt wird. Sie könnten beispielsweise ein System haben, das eine große Anzahl von Verbindungen anfordert, jedoch nur über eine begrenzte Speicherkapazität zur Verarbeitung dieser Verbindungen verfügt. In diesem Fall kann die Anpassung dieses Parameters sehr nützlich sein, da es hier aufgrund hoher gleichzeitiger Aktivität zu übermäßigem Seitenwechseln durch das Betriebssystem kommen kann.

Der Wert -1 gibt an, dass der Grenzwert gleich dem Wert des Parameters **max_coordagents** ist.

Empfehlung: In den meisten Fällen kann der Standardwert für diesen Parameter übernommen werden. In Fällen, in denen es durch den gleichzeitigen Zugriff zahlreicher Anwendungen zu Problemen kommt, können Sie durch Vergleichstests (Benchmark-Tests) die beste Einstellung für diesen Parameter ermitteln, um die Leistung der Datenbank zu optimieren.

mon_heap_sz - Zwischenspeichergröße für Datenbanksystemmonitor

Mit diesem Parameter wird der Speicherbereich in Seiten festgelegt, der für Daten des Datenbanksystemmonitors zugeordnet werden soll. Der Speicher wird aus dem Monitorzwischenspeicher zugeordnet, wenn Sie eine Datenbankmonitorfunktion ausführen (z. B. Monitorschalter aktivieren, Monitordaten zurücksetzen, Ereignismonitor aktivieren oder Monitorereignisse an einen aktiven Ereignismonitor senden).

Ab Version 9.5 hat dieser Datenbankkonfigurationsparameter den Standardwert AUTOMATIC, was bedeutet, dass der Monitorzwischenspeicher nach Bedarf erhöht wird, bis der Grenzwert des Parameters **instance_memory** erreicht ist.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
`Automatic [0 - 60 000]`

Maßeinheit
Seiten (4 KB)

Zuordnung
Wenn der Datenbankmanager mit dem Befehl **db2start** gestartet wird

Freigabe
Wenn der Datenbankmanager mit dem Befehl **db2stop** gestoppt wird

Erhält der Parameter den Wert 0, kann der Datenbankmanager keine Daten des Datenbanksystemmonitors sammeln.

Empfehlung: Die für die Monitorfunktionen erforderliche Speicherkapazität ist von der Anzahl der Überwachungsanwendungen (Anwendungen, die Momentaufnahmen machen, oder Ereignismonitoren), den aktivierten Schaltern und der Datenbankaktivität abhängig.

Wenn die konfigurierte Speicherkapazität in diesem Zwischenspeicher erschöpft und kein weiterer nicht reservierter Speicher mehr verfügbar ist, wird eines der folgenden Ereignisse eintreten:

- Wenn die erste Anwendung eine Verbindung zu der Datenbank herstellt, für die dieser Ereignismonitor definiert ist, wird eine Fehlernachricht in das Benachrichtigungsprotokoll für die Verwaltung geschrieben.
- Wenn ein Ereignismonitor fehlschlägt, der mithilfe der Anweisung SET EVENT MONITOR dynamisch gestartet wird, wird ein Fehlercode an die Anwendung zurückgegeben.
- Wenn ein Monitorbefehl oder eine API-Unterroutine fehlschlägt, wird ein Fehlercode an die Anwendung zurückgegeben.
- Wenn eine Anwendung einen Ereignismonitordatensatz senden muss, der jedoch nicht aus dem Monitorzwischenspeicher zugeordnet werden kann, wird die Anwendung möglicherweise blockiert, bis ein Datensatz zugeordnet werden kann.

nodetype - Knotentyp der Instanz ()

Dieser Parameter gibt den Instanztyp (Datenbankmanagertyp) an, der von DB2-Produkten erstellt wird, die auf Ihrer Maschine installiert sind.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Informativ

Die Werte, die von diesem Parameter zurückgegeben werden können, sowie die Produkte, die dem betreffenden Knotentyp zugeordnet sind, werden im Folgenden beschrieben:

- **Datenbankserver mit lokalen und fernen Clients:** Ein DB2-Serverprodukt, das lokale und ferne Data Server Runtime Client-Systeme unterstützt und auf andere ferne Datenbankserver zugreifen kann.
- **Client:** Ein Data Server Runtime Client, der auf ferne Datenbankserver zugreifen kann.
- **Datenbankserver mit lokalen Clients:** Ein Verwaltungssystem für relationale DB2-Datenbanken, das lokale Data Server Runtime Client-Systeme unterstützt und auf andere ferne Datenbankserver zugreifen kann.
- **Partitionierter Datenbankserver mit lokalen und fernen Clients:** Ein DB2-Serverprodukt, das lokale und ferne Data Server Runtime Client-Systeme unterstützt, auf andere ferne Datenbankserver zugreifen kann und parallelitätsfähig ist.

notifylevel - Benachrichtigungsstufe

Dieser Parameter gibt den Typ der Hinweisnachrichten zur Systemverwaltung an, die in das Protokoll mit Benachrichtigungen für die Systemverwaltung geschrieben werden.

Konfigurationstyp
Datenbankmanager

Gilt für
- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
`3 [ 0 — 4 ]`

Unter Linux- und UNIX-Betriebssystemen ist das Protokoll mit Benachrichtigungen für die Systemverwaltung eine Textdatei mit dem Namen *instanz*`.nfy`. Unter Windows werden alle Hinweisnachrichten für die Systemverwaltung in das Ereignisprotokoll geschrieben. Die Fehler können von DB2, dem Diagnosemonitor, den Capture- und Apply-Programmen sowie von Benutzeranwendungen geschrieben werden.

Gültige Werte für diesen Parameter sind:
- `0` — Keine Aufzeichnung von Hinweisnachrichten zur Systemverwaltung. (Diese Einstellung wird nicht empfohlen.)
- `1` — Schwer wiegende oder nicht behebbare Fehler. Es werden nur schwer wiegende und nicht behebbare Fehler protokolliert. Zur Behebung einiger dieser Fehler benötigen Sie möglicherweise Unterstützung vom DB2-Service.
- `2` — Sofortmaßnahmen erforderlich. Es werden Bedingungen protokolliert, die Sofortmaßnahmen durch den Systemadministrator oder den Datenbankadministrator erfordern. Wenn die Bedingung nicht behoben wird, könnte dies zu einem schwer wiegenden Fehler führen. Auf dieser Ebene können ebenfalls äußerst wichtige, nicht fehlerbezogene Aktivitäten (wie zum Beispiel eine Recovery) pro-

tokolliert werden. Auf dieser Ebene werden Alarmnachrichten des Diagnosemonitors erfasst. Informationsnachrichten werden auf dieser Ebene angezeigt.

- 3 — Wichtige Informationen, keine Sofortmaßnahmen erforderlich. Es werden Bedingungen protokolliert, die nicht bedrohlich sind und keine Sofortmaßnahmen erfordern, die jedoch auf ein nicht optimales System hinweisen können. Auf dieser Ebene werden Alarmnachrichten, Warnungen und Informationsnachrichten des Diagnosemonitors erfasst.

Hinweise

- Das Protokoll mit Benachrichtigungen für die Systemverwaltung enthält Nachrichten mit Werten bis einschließlich dem Wert von **notifylevel**. Wenn der Parameter **notifylevel** beispielsweise auf 3 gesetzt ist, enthält dieses Protokoll Nachrichten der Ebenen 1, 2 und 3.
- Eine Benutzeranwendung muss die API db2AdminMsgWrite aufrufen, um in die Benachrichtigungsdatei oder das Windows-Ereignisprotokoll schreiben zu können.
- Es kann hilfreich sein, den Wert dieses Parameters zu erhöhen, um zusätzliche Fehlerbestimmungsdaten zu sammeln, die zur Lösung eines Problems beitragen können. Wenn der Diagnosemonitor Benachrichtigungen an die in seiner Konfiguration angegebenen Ansprechpartner senden soll, müssen Sie für den Parameter **notifylevel** den Wert 2 oder höher angeben.
- In bestimmten Situationen überschreibt DB2 die Einstellung des Parameters **notifylevel**, um Nachrichten mit hohem Stellenwert anzuzeigen.

num_initagents - Anfangswert für die Anzahl Agenten im Pool

Dieser Parameter gibt an, wie viele inaktive Agenten beim Ausführen von **db2start** im Agentenpool erstellt werden.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
0 [0–64 000]

Anmerkung: Der Standardwert unterliegt nach der ursprünglichen Datenbankerstellung Änderungen durch den DB2 Configuration Advisor.

Der Datenbankmanager startet immer die inaktiven **num_initagents**-Agenten im Rahmen des Befehls **db2start**, es sei denn, der Wert dieses Parameters ist beim Start größer als **num_poolagents** und **num_poolagents** ist nicht auf AUTOMATIC gesetzt. In diesem Fall startet der Datenbankmanager nur die inaktiven **num_poolagents**-Agenten, da es keinen Grund gibt, mehr inaktive Agenten zu starten als in einen Pool gestellt werden können.

num_initfenced - Anfangswert für die Anzahl abgeschirmter Prozesse

Dieser Parameter gibt die Anfangsanzahl der inaktiven **db2fmp**-Prozesse an, die keine Threadprozesse sind und die zum Startzeitpunkt des Datenbankmanagers (**START DBM**) im Pool **db2fmp** erstellt werden.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
0 [0–64 000]

Wenn Sie diesen Parameter festlegen, wird die einleitende Startzeit für das Ausführen von nicht threadsicheren C- und COBOL-Routinen verringert. Dieser Parameter wird ignoriert, wenn **keepfenced** nicht angegeben wird.

Es ist viel wichtiger, den Parameter **fenced_pool** auf eine für Ihr System geeignete Größe festzulegen, als zum Startzeitpunkt des Datenbankmanagers (**START DBM**) eine Reihe von **db2fmp**-Prozessen zu starten.

In früheren Versionen hatte dieser Parameter den Namen **num_initdaris**.

num_poolagents - Agentenpoolgröße

Dieser Parameter definiert die maximale Größe des Pools inaktiver Agenten.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Standardwert
100, AUTOMATIC [-1, 0 - 64000]

Anmerkung: Der Standardwert unterliegt nach der ursprünglichen Datenbankerstellung Änderungen durch den DB2 Configuration Advisor.

Dieser Konfigurationsparameter ist auf AUTOMATIC mit dem Standardwert 100 gesetzt. Die Einstellung -1 wird unterstützt und in den Wert 100 umgesetzt. Wenn dieser Parameter auf AUTOMATIC gesetzt ist, verwaltet der Datenbankmanager automatisch die Anzahl inaktiver Agenten, die in einem Pool zusammengefasst werden sollen. In der Regel bedeutet dies Folgendes: Wenn ein Agent seine Arbeit beendet, wird nicht der Agent selbst beendet, sondern er ist nur für einen bestimmten

Zeitraum inaktiv. In Abhängigkeit von der Auslastung und dem Typ des Agenten, wird er möglicherweise nach einer bestimmten Zeit beendet.

Wenn Sie AUTOMATIC verwenden, können Sie weiterhin einen Wert für den Konfigurationsparameter **num_poolagents** angeben. Weitere inaktive Agenten werden immer dann in einem Pool zusammengefasst, wenn die aktuelle Anzahl von inaktiven Agenten, die in einem Pool zusammengefasst sind, kleiner-gleich dem von Ihnen angegebenen Wert ist.

Beispiele

num_poolagents ist auf 100 und AUTOMATIC gesetzt
Wenn ein Agent frei wird, wird er dem Pool inaktiver Agenten hinzugefügt; dort prüft der Datenbankmanager zu einem bestimmten Zeitpunkt, ob er beendet werden sollte. Wenn zu dem Zeitpunkt, zu dem der Datenbankmanager die Beendigung des Agenten in Betracht zieht, die Gesamtzahl der inaktiven Agenten, die in einem Pool zusammengefasst sind, größer als 100 ist, wird dieser Agent beendet. Wenn weniger als 100 inaktive Agenten vorhanden sind, wartet der inaktive Agent weiterhin auf Arbeit. Durch die Verwendung der Einstellung AUTOMATIC können weitere inaktive Agenten (über 100) in einem Pool zusammengefasst werden. Dies ist möglicherweise in Zeiten größerer Systemaktivität nützlich, wenn die Häufigkeit von Arbeit in größerem Umfang schwanken kann. In den Fällen, in denen höchstwahrscheinlich weniger als 100 inaktive Agenten zu einer beliebigen Zeit auftreten, werden Agenten garantiert in einem Pool zusammengefasst. Zu Zeiten von weniger Systemaktivität kann dies positiv genutzt werden; der Initialisierungsaufwand für neue Arbeit ist dann geringer.

num_poolagents wird dynamisch konfiguriert
Wenn der Wert des Parameters auf einen Wert erhöht wird, der die Anzahl der in einem Pool zusammengefassten Agenten überschreitet, wirkt sich diese Änderung unmittelbar aus. Wenn neue Agenten inaktiviert werden, werden sie in einem Pool zusammengefasst. Wenn der Wert des Parameters verringert wird, verkleinert der Datenbankmanager die Anzahl der Agenten in dem Pool nicht unmittelbar. Stattdessen bleibt die Größe des Pools bestehen und Agenten werden nach der Verwendung beendet und werden wieder inaktiv; dadurch wird die Anzahl der Agenten in dem Pool schrittweise auf den neuen Grenzwert reduziert.

Empfehlung

Für die meisten Umgebungen sind die Standardwerte 100 und AUTOMATIC ausreichend. Wenn Sie bei einer bestimmten Auslastung das Gefühl haben, dass zu viele Agenten erstellt und beendet werden, können Sie den Wert von **num_poolagents** erhöhen. Der Parameter sollte allerdings auf AUTOMATIC gesetzt bleiben.

numdb - Maximale Anzahl gleichzeitig aktiver Datenbanken einschließlich Host- und System i-Datenbanken

Dieser Parameter gibt die Anzahl der lokalen Datenbanken, die gleichzeitig aktiv sein können, oder die maximale Anzahl verschiedener Datenbankaliasnamen an, die in einem DB2 Connect-Server katalogisiert werden können.

Dieser Parameter ist überladen.

Die Wirkung des Konfigurationsparameters **numdb** des Datenbankmanagers ist von der Umgebung abhängig, in der er verwendet wird.

- In einer DB2 Connect-Umgebung können Sie die maximale Anzahl der Datenbankaliasnamen, die auf einem DB2 Connect-Server katalogisiert werden können, mit dem Konfigurationsparameter **numdb** des Datenbankmanagers angeben.
- Außerhalb einer DB2 Connect-Umgebung können Sie die Anzahl der lokalen Datenbanken, die gleichzeitig aktiv sein können, mit dem Konfigurationsparameter **numdb** des Datenbankmanagers angeben.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver nur mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]

UNIX-DB2 pureScale-Umgebung
32 [1 — 200]

UNIX-Datenbankserver mit lokalen und fernen Clients
32 [1 — 256]

UNIX-Datenbankserver nur mit lokalen Clients
8 [1 — 256]

Windows-Datenbankserver mit lokalen und fernen Clients
32 [1 — 256]

Windows-Datenbankserver nur mit lokalen Clients
3 [1 — 256]

Maßeinheit
Zähler

Hinweise zur Verwendung

- Beim Festlegen oder Aktualisieren dieses Parameters ist unbedingt zu berücksichtigen, dass jede Datenbank Speicherplatz belegt und jede aktive Datenbank ein zusätzliches neues Segment im gemeinsam genutzten Speicher erfordert.
- Setzen Sie **numdb** auf den Standardwert, wenn die Anzahl der auf der Instanz ausgeführten Datenbanken kleiner als der Standardwert ist.
- Die Verwendung des Konfigurationsparameters **numdb** muss mit den Konfigurationsparametern **db2_database_cf_memory** und **cf_db_mem_sz** abgeglichen werden.
- Eine Änderung des Parameters **numdb** kann Auswirkungen auf die Gesamtmenge an Speicher haben, der zugeordnet wird. Daher ist es nicht empfehlenswert, diesen Parameter häufig zu aktualisieren. Planen Sie die Konfiguration für alle Parameter, die mit der Speicherzuordnung für eine Datenbank (oder für eine mit dieser Datenbank verbundene Anwendung) in Zusammenhang stehen im Voraus, und ändern Sie all diese Parameter zur gleichen Zeit.
- Neben dem Grenzwert **numdb** in einer DB2 pureScale-Umgebung besteht ebenfalls ein Höchstwert für die Anzahl der Datenbankaktivierungen für alle Member in einer Instanz. Die maximale Anzahl beträgt 512 Datenbankaktivierungen. Beispiel: Wenn jedes Member in einer DB2 pureScale-Umgebung mit vier

Membern 200 Datenbanken aktiviert, bedeutet dies eine Gesamtzahl von 800 Datenbankaktivierungen durch Member. Da die Zahl von 800 Datenbankaktivierungen den maximalen Wert überschreitet, wird ein Fehler zurückgegeben.

- Wenn in einer Umgebung mit mehreren Datenbanken nach dem Absturz eines Members eine Recovery erforderlich wird, wird die Anzahl der Datenbanken, die auf jedem Member parallel zueinander wiederhergestellt werden, durch den Wert des Konfigurationsparameters **numdb** oder der Registrierdatenbankvariable **DB2_MCR_RECOVERY_PARALLELISM_CAP** definiert, je nachdem, welcher dieser beiden Werte kleiner ist.

query_heap_sz - Größe des Abfragezwischenspeichers

Dieser Parameter gibt die maximale Speichermenge an, die für den Abfragezwischenspeicher zugeordnet werden kann. Dabei wird sichergestellt, dass eine Anwendung nicht unnötig große virtuelle Speicherbereiche innerhalb eines Agenten belegt.

Wichtig: Dieser Parameter gilt in Version 9.5 als veraltet und wird möglicherweise in einem zukünftigen Release entfernt. Dieser Parameter kann für Datenserver und Clients einer Version vor Version 9.5 weiterhin verwendet werden. In Version 9.5 und späteren Releases wird der für diesen Konfigurationsparameter angegebene Wert ignoriert.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
`1 000 [2 - 524 288 ]`

Maßeinheit
Seiten (4 KB)

Zuordnung
Wenn eine Anwendung (lokal oder fern) eine Verbindung zur Datenbank herstellt

Freigabe
Wenn die Anwendung die Verbindung zur Datenbank oder zur Instanz trennt

Ein Abfragezwischenspeicher wird zur Speicherung jeder Abfrage im privaten Speicher des Agenten verwendet. Die Informationen für jede Abfrage bestehen aus dem Eingabe- und Ausgabe-SQL-Deskriptorbereich, dem Text der Anweisung, dem SQL-Kommunikationsbereich, dem Paketnamen, dem Ersteller, der Abschnittsnummer und dem Konsistenz-Token.

Aus dem Abfragezwischenspeicher wird auch der Speicher für Blockcursor zugeordnet. Dieser Speicher besteht aus einem Cursorsteuerungsblock und einem vollständig formatierten Ausgabe-SQL-Deskriptorbereich.

Zu Anfang ist der zugeordnete Abfragezwischenspeicher ebenso groß wie der Zwischenspeicher für die Anwendungsunterstützungsebene, der durch den Parameter **aslheapsz** definiert ist. Der Abfragezwischenspeicher muss größer oder gleich 2 sowie größer oder gleich dem Wert des Parameters **aslheapsz** sein. Wenn dieser Abfragezwischenspeicher zur Verarbeitung einer Anforderung nicht ausreicht, wird neuer Speicher in der für die Anforderung erforderlichen Größe (nicht über den Wert **query_heap_sz** hinaus) zugeordnet. Wenn dieser neue Abfragezwischenspeicher mehr als anderthalbmal so groß wie der Wert für **aslheapsz** ist, wird der Abfragezwischenspeicher in der Größe von **aslheapsz** neu zugeordnet, wenn die Abfrage beendet ist.

Empfehlung: In den meisten Fällen ist der Standardwert ausreichend. Als Minimalwert sollte für **query_heap_sz** ein Wert angegeben werden, der mindestens fünfmal so groß wie der Wert für **aslheapsz** ist. Dadurch können Abfragen größer als **aslheapsz** sein, und es wird zusätzlicher Speicher für drei oder vier Blockcursor bereitgestellt, die gleichzeitig geöffnet sein können.

Wenn Sie sehr umfangreiche LOBs (große Objekte) haben, müssen Sie möglicherweise den Wert dieses Parameters erhöhen, damit der Abfragezwischenspeicher groß genug für diese LOBs ist.

release - Release-Level der Datenbankkonfigurationsdatei

Dieser Parameter gibt den Release-Level der Konfigurationsdatei an.

Konfigurationstyp
Datenbankmanager, Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Informativ

rstrt_light_mem - Speicher für Light-Neustart (Konfigurationsparameter)

Dieser Parameter gibt die maximale Speichermenge an, die auf einem Host für Light-Neustartoperationen zugeordnet und reserviert wird. Die Speichermenge wird als Prozentsatz des Konfigurationsparameters **instance_memory** angegeben.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
`AUTOMATIC` [1 - 10 für `AUTOMATIC`; der benutzerdefinierte Bereich ist `1 - 50`]

Maßeinheit
Prozentsatz des Instanzspeichers

Zuordnung
Wenn die Instanz gestartet wird

Freigabe
Wenn die Instanz gestoppt wird

Wenn der Parameter **rstrt_light_mem** auf AUTOMATIC gesetzt ist, berechnet der DB2-Datenbankmanager beim Starten einer Instanz automatisch eine feste obere Grenze für die Speichermenge, die vorab zugeordnet und für Zwecke der Recovery mit Light-Neustart reserviert wird. Die angegebene Speichermenge wird reserviert, um fehlgeschlagene Member aufzunehmen, die auf einem anderen als dem zugehörigen Home-Host erneut gestartet werden müssen (Modus für Light-Neustart). Der DB2-Datenbankmanager berechnet den Wert entsprechend den aktuellen Einstellungen der Parameter **instance_memory** und **numdb** (maximale Anzahl der derzeit aktiven Datenbanken) sowie unter Berücksichtigung der Anzahl Member auf dem Host. Der automatisch berechnete Wert beträgt zwischen 1 und 10 Prozent der Instanzspeicherbegrenzung und ist in der Gesamtgröße des Instanzspeichers enthalten. Die Benutzer können den Parameter **rstrt_light_mem** auch explizit auf einen Wert zwischen 1 und 50 Prozent der Instanzspeicherbegrenzung setzen. Zusätzlicher Speicher könnte die Recoveryzeit verkürzen, insbesondere wenn für mehrere Datenbanken eine Recovery erforderlich ist. Durch zusätzlichen Speicher würde aber auch der verfügbare Speicher für die normale Arbeit verringert und damit der Durchsatz reduziert, der von den Membern verarbeitet werden kann.

Der Parameter kann online konfiguriert werden. Nachdem der reservierte Recoveryspeicher beim Starten der DB2-Instanz zugeordnet wurde, ist die Speichermenge festgelegt und kann nur geändert werden, indem der Konfigurationswert geändert und die DB2-Instanz global gestoppt und erneut gestartet wird.

Informationen zur gesamten Speichermenge, die auf einem Host zugeordnet ist, können Sie mit dem Befehl **db2pd** unter Angabe der neuen Option **-totalmem** aufrufen. Diese Informationen sind auch in der Angabe über die auf einem Host zugeordnete Speichermenge für den Light-Neustart enthalten. Es werden nur Informationen zu dem Host zurückgegeben, auf den derzeit zugegriffen wird. Der Befehl **db2pd** kann jedoch parallel auf mehreren Hosts ausgeführt werden.

resync_interval - Intervall für Transaktionsresynchronisation

Mit diesem Parameter wird das Zeitintervall in Sekunden angegeben, während dessen ein Transaktionsmanager (TM), ein Ressourcenmanager (RM) oder ein Synchronisationspunktmanager (SPM) versuchen soll, jede ausstehende unbestätigte Transaktion, die in dem TM, RM oder SPM gefunden wird, wiederherzustellen.

Dieser Parameter ist gültig, wenn Sie Transaktionen in einer DUOW-Umgebung (DUOW = Distributed Unit of Work, verteilte Arbeitseinheit) ausführen. Dieser Parameter gilt ebenfalls für die Recovery von Systemen mit föderierten Datenbanken.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
180 [1 - 60 000]

Maßeinheit
Sekunden

Empfehlung: Wenn in Ihrer Umgebung unbestätigte Transaktionen keine Störungen anderer Transaktionen für Ihre Datenbank verursachen, können Sie den Wert dieses Parameters auch erhöhen. Wenn Sie ein DB2 Connect-Gateway zum Zugriff auf DRDA2-Anwendungsserver verwenden, sollten Sie die Auswirkungen bedenken, die unbestätigte Transaktionen auf Anwendungsservern haben könnten, auch wenn lokal keine Störungen beim Datenzugriff zu erwarten sind. Gibt es keine unbestätigten Transaktionen, sind die Auswirkungen auf die Leistung minimal.

rqrioblk - E/A-Blockgröße für Clients

Dieser Parameter gibt die Blockgröße auf dem Data Server Runtime Client an, wenn ein Blockcursor geöffnet wird.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit fernen Clients
- Client
- Partitionierten Datenbankserver mit fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
32 767 [4 096 - 65 535]

Maßeinheit
Byte

Der Speicher für Blockcursor wird aus dem privaten Adressraum der Anwendung zugeordnet. Sie sollten daher die optimale Größe des privaten Speichers ermitteln, der jedem Anwendungsprogramm zugeordnet werden soll. Wenn der Data Server Runtime Client keinen Bereich für einen Blockcursor aus dem privaten Speicher der Anwendung zuordnen kann, wird ein Cursor ohne Blockung geöffnet.

Beachten Sie außerdem, welche Auswirkung dieser Parameter auf die Anzahl und die mögliche Größe von Blockcursorn hat. Große Zeilenblöcke können zu einer höheren Leistung führen, wenn die Anzahl oder die Größe der Zeilen, die übertragen werden, groß ist (z. B., wenn die Datenmenge größer als 4.096 Byte ist). Dies hat jedoch den Nachteil, dass größere Datensatzblöcke die Größe des für jede Verbindung benötigten Arbeitsspeichers erhöhen.

Größere Satzblöcke können außerdem mehr Abrufanforderungen verursachen, als tatsächlich für die Anwendung erforderlich wären. Sie können die Anzahl der Abrufanforderungen mit der Klausel OPTIMIZE FOR in der Anweisung SELECT in Ihrer Anwendung steuern.

sheapthres - Schwellenwert für Sortierspeicher

Dieser Parameter ein instanzweiter, veränderlicher Grenzwert für den maximalen Speicherbereich, der zu einem beliebigen Zeitpunkt von privaten Sortiervorgängen beansprucht werden kann. Wenn die Gesamtnutzung an privatem Sortierspeicher-

bereich für eine Instanz diesen Grenzwert erreicht, wird der für weitere eingehende private Sortieranforderungen zugeordnete Speicherbereich beträchtlich verkleinert.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients
- OLAP-Funktionen

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]

UNIX-Plattformen (32-Bit)
`0 [0, 250 - 2097152]`

Windows-Plattformen (32-Bit)
`0 [0, 250 - 2097152]`

64-Bit-Plattformen
`0 [0, 250 - 2147483647]`

Anmerkung: Der Standardwert unterliegt nach der ursprünglichen Datenbankerstellung Änderungen durch den DB2 Configuration Advisor.

Maßeinheit
Seiten (4 KB)

Der Sortierspeicher wird beispielsweise bei folgenden Operationen verwendet: Sortierungen, Hash-Joins, dynamische Bitzuordnungen (die für logisches Verknüpfen von Indizes über AND (Index ANDing) und Sternjoins verwendet werden) und Operationen, bei denen sich die Tabelle im Speicher befindet.

Durch die explizite Angabe des Schwellenwerts wird verhindert, dass der Datenbankmanager übermäßig große Speichermengen für sehr viele Sortiervorgänge verwendet.

Es gibt keinen Grund, den Wert dieses Parameter beim Versetzen von einer Umgebung mit nicht partitionierten Datenbanken in eine Umgebung mit partitionierten Datenbanken zu erhöhen. Wenn Sie die Konfigurationsparameter der Datenbank und des Datenbankmanagers in einer Umgebung mit einer nicht partitionierten Datenbank optimiert haben, funktionieren diese Werte in einer Umgebung mit partitionierten Datenbanken zumeist ebenfalls einwandfrei. Sie können diesen Parameter in anderen Knoten oder Datenbankpartitionen nur auf unterschiedliche Werte setzen, indem Sie mehrere DB2-Instanzen erstellen. Dies erfordert das Verwalten verschiedener DB2-Datenbanken in verschiedenen Datenbankpartitionsgruppen. Ein solches Vorgehen macht viele Vorteile einer Umgebung mit partitionierten Datenbanken zunichte.

Wenn der Parameter **sheapthres** der Instanzebene auf den Wert `0` gesetzt ist, erfolgt die Verfolgung der Sortierspeichernutzung nur auf der Datenbankebene und

die Speicherzuordnung für Sortiervorgänge wird durch den Wert des Konfigurationsparameters **sheapthres_shr** auf der Datenbankebene begrenzt.

Die automatische Optimierung von **sheapthres_shr** ist nur zulässig, wenn der Konfigurationsparameter **sheapthres** des Datenbankmanagers auf den Wert `0` gesetzt ist.

Dieser Parameter ist nicht dynamisch aktualisierbar, wenn eine der folgenden Bedingungen zutrifft:

- Der Anfangswert für **sheapthres** ist `0` und der Zielwert ist ungleich 0.
- Der Anfangswert für **sheapthres** ist ungleich 0, und der Zielwert ist 0.

Empfehlung: Im Idealfall sollten Sie den Wert dieses Parameters möglichst auf ein angemessenes Vielfaches des Parameters **sortheap** mit dem größten Wert setzen, der in Ihrer Datenbankmanagerinstanz definiert ist. Der Wert dieses Parameters sollte **mindestens** zweimal so groß sein wie der größte Sortierspeicher (**sortheap**), der für eine Datenbank in der Instanz definiert ist.

Wenn Sie private Sortiervorgänge ausführen und Ihr System über genügend Speicher verfügt, kann der Idealwert für diesen Parameter auf folgende Weise berechnet werden:

1. Berechnen Sie die normale Sortierspeicherbelegung für jede Datenbank:

   ```
   (normale Anzahl Agenten, die gleichzeitig für die Datenbank ausgeführt werden)
    * (sortheap, wie für die Datenbank definiert)
   ```

2. Berechnen Sie die Summe der obigen Ergebnisse. Dies ergibt einen Wert für den gesamten Sortierspeicher, der unter normalen Umständen für alle Datenbanken innerhalb der Instanz verwendet werden kann.

Sie sollten Vergleichstests (Benchmark-Tests) ausführen, um die optimale Einstellung für diesen Parameter zu ermitteln, die Sortierleistung und Speicherbedarf gleichermaßen berücksichtigt.

Mithilfe des Datenbanksystemmonitors können Sie unter Verwendung des Monitorelements **post_threshold_sorts** (Sortierungen nach Erreichen des Schwellenwerts) die Sortieraktivitäten verfolgen.

spm_log_file_sz - Protokolldateigröße für SPM

Mit diesem Parameter wird die Größe der Protokolldatei für den Synchronisationspunktmanager (SPM) in 4-KB-Seiten angegeben.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
`256 [4 - 1000]`

Maßeinheit
Seiten (4 KB)

Die Protokolldatei befindet sich im Unterverzeichnis spmlog des Verzeichnisses sqllib und wird erstellt, wenn der Synchronisationspunktmanager (SPM) zum ersten Mal gestartet wird.

Empfehlung: Die Protokolldatei des Synchronisationspunktmanagers sollte einerseits groß genug sein, um die Leistung zu gewährleisten, andererseits aber auch klein genug, um die Verschwendung von Speicherbereich zu vermeiden. Die erforderliche Größe hängt von der Anzahl der Transaktionen ab, die geschützte Dialoge verwenden, und von der Häufigkeit, mit der Commits oder Rollbacks ausgeführt werden.

Gehen Sie wie folgt vor, um die Größe der SPM-Protokolldatei zu ändern:

1. Verwenden Sie den Befehl **LIST DRDA INDOUBT TRANSACTIONS** , um festzustellen, ob es unbestätigte Transaktionen gibt.
2. Wenn es keine unbestätigten Transaktionen gibt, stoppen Sie den Datenbankmanager.
3. Aktualisieren Sie die Konfiguration des Datenbankmanagers mit dem neuen Wert für die Größe der SPM-Protokolldatei.
4. Wechseln Sie in das Verzeichnis $HOME/sqllib, und löschen Sie die aktuelle SPM-Protokolldatei mit dem Befehl rm -fr spmlog. (Anmerkung: Dies ist der AIX-Befehl. Auf anderen Systemen sind wahrscheinlich andere Löschbefehle erforderlich.)
5. Starten Sie den Datenbankmanager. Beim Start des Datenbankmanagers wird eine neue SPM-Protokolldatei in der angegebenen Größe erstellt.

spm_log_path - Protokolldateipfad für SPM

Dieser Parameter gibt an, in welches Verzeichnis die SPM-Protokolle geschrieben werden.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
NULL [beliebiger gültiger Pfad oder gültige Einheit]

Wenn dieser Parameter null ist, werden die Protokolle standardmäßig in das Verzeichnis sqllib/spmlog geschrieben. Dies kann in Umgebungen mit hohem Transaktionsaufkommen zu E/A-Engpässen führen. Verwenden Sie diesen Parameter, damit SPM-Protokolldateien auf eine Platte mit kürzeren Zugriffszeiten als das aktuelle Verzeichnis sqllib/spmlog geschrieben werden. Dadurch wird der gemeinsame Zugriff der SPM-Agenten optimiert.

Anmerkung: Wenn der SPM aktiviert ist, wird das Standardverzeichnis erstellt, wenn es noch nicht vorhanden ist. Zum Aktivieren des SPMs muss der Konfigurationsparameter **spm_name** festgelegt werden.

spm_max_resync - Maximale Anzahl von SPM-Resynchronisationsagenten

Mit diesem Parameter wird die Anzahl der Agenten angegeben, die gleichzeitig Resynchronisationsoperationen ausführen können.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
20 [10 — 256]

spm_name - Name des Synchronisationspunktmanagers

Mit diesem Parameter wird der Name der Instanz des Synchronisationspunktmanagers (SPM) gegenüber dem Datenbankmanager identifiziert.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert
Vom TCP/IP-Hostnamen abgeleitet

srvcon_auth - Authentifizierungstyp für ankommende Verbindungen auf dem Server

Mit diesem Parameter wird angegeben, wie und wo die Benutzerauthentifizierung bei der Verarbeitung ankommender Verbindungen auf dem Server stattfinden soll. Er dient zum Überschreiben des aktuellen Authentifizierungstyps.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
`NOT_SPECIFIED [NOT_SPECIFIED; CLIENT; SERVER; SERVER_ENCRYPT; DATA_ENCRYPT; DATA_ENCRYPT_CMP; KERBEROS; KRB_SERVER_ENCRYPT; GSSPLUGIN; GSS_SERVER_ENCRYPT]`

Wenn kein Wert angegeben wird, verwendet DB2 den Wert des Konfigurationsparameters **authentication** des Datenbankmanagers.

Eine Beschreibung der Authentifizierungstypen finden Sie unter „authentication - Authentifizierungstyp" auf Seite 759.

srvcon_gssplugin_list - Liste der GSS-API-Plug-ins für ankommende Verbindungen auf dem Server

Mit diesem Parameter werden die GSS-API-Plug-in-Bibliotheken angegeben, die vom Datenbankserver unterstützt werden. Dieser Parameter verarbeitet ankommende Verbindungen auf dem Server, wenn für den Parameter **srvcon_auth** der Wert KERBEROS, KRB_SERVER_ENCRYPT, GSSPLUGIN oder GSS_SERVER_ENCRYPT angegeben ist oder wenn **srvcon_auth** nicht angegeben ist und für **authentication** der Wert KERBEROS, KRB_SERVER_ENCRYPT, GSSPLUGIN oder GSS_SERVER_ENCRYPT angegeben ist.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
NULL [beliebige gültige Zeichenfolge]

Dieser Wert ist standardmäßig null. Wenn der Authentifizierungstyp GSSPLUGIN ist und dieser Parameter NULL ist, wird ein Fehler zurückgegeben. Wenn der Authentifizierungstyp KERBEROS ist und dieser Parameter NULL ist, wird das von DB2 bereitgestellte Kerberos-Modul bzw. die Kerberos-Bibliothek verwendet. Bei einem anderen Authentifizierungstyp wird dieser Parameter nicht verwendet.

Wenn der Authentifizierungstyp KERBEROS ist und der Wert dieses Parameters nicht NULL ist, muss die angegebene Liste exakt ein Kerberos-Plug-in enthalten, und dieses Plug-in wird zur Authentifizierung verwendet (alle anderen GSS-Plug-ins in der Liste werden ignoriert). Enthält die Liste mehr als ein Kerberos-Plug-in, wird ein Fehler zurückgegeben.

Jeder Name eines GSS-API-Plug-ins muss durch ein Komma (,) ohne Leerzeichen vor oder nach dem Komma getrennt werden. Die Plug-in-Namen sollten in der bevorzugten Reihenfolge aufgelistet werden.

srvcon_pw_plugin - Plug-in für Benutzer-ID/Kennwort für ankommende Verbindungen auf dem Server

Mit diesem Parameter wird der Name der Benutzer-ID/Kennwort-Plug-in-Standardbibliothek angegeben, die für die serverseitige Authentifizierung zu verwenden ist.

Er verarbeitet ankommende Verbindungen auf dem Server, wenn der Parameter **srvcon_auth** mit dem Wert CLIENT, SERVER, SERVER_ENCRYPT, DATA_ENCRYPT oder

DATA_ENCRYPT_CMP angegeben ist oder wenn **srvcon_auth** nicht angegeben ist und **authentication** mit dem Wert CLIENT, SERVER, SERVER_ENCRYPT, DATA_ENCRYPT oder DATA_ENCRYPT_CMP angegeben ist.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
NULL [beliebige gültige Zeichenfolge]

Standardmäßig ist dieser Wert null und die von der DB2-Datenbank bereitgestellte Benutzer-ID/Kennwort-Plug-in-Bibliothek wird verwendet. Für Nichtrootinstallationen gilt, dass bei Verwendung der DB2-Bibliothek des Plug-ins für Benutzer-ID und Kennwort der Befehl **db2rfe** ausgeführt werden muss, bevor das DB2-Datenbankprodukt verwendet wird.

srv_plugin_mode - Server-Plug-in-Modus

Mit diesem Parameter wird angegeben, ob Plug-ins im abgeschirmten oder nicht abgeschirmten Modus auszuführen sind. Es wird nur der nicht abgeschirmte Modus (UNFENCED) unterstützt.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
UNFENCED

ssl_cipherspecs - Unterstützte Verschlüsselungsspezifikationen auf dem Server (Konfigurationsparameter)

Mit diesem Konfigurationsparameter werden die Cipher Suites angegeben, die der Server für ankommende Verbindungsanforderungen bei Verwendung des SSL-Protokolls zulässt.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
Null [TLS_RSA_WITH_AES_256_CBC_SHA; TLS_RSA_WITH_AES_128_CBC_SHA; TLS_RSA_WITH_3DES_EDE_CBC_SHA]

Sie können mehrere Verschlüsselungsspezifikationen angeben, wie zum Beispiel TLS_RSA_WITH_AES_256_CBC_SHA oder TLS_RSA_WITH_AES_128_CBC_SHA oder TLS_RSA_WITH_3DES_EDE_CBC_SHA. Diese Angaben müssen durch ein Komma (,) ohne Leerzeichen vor oder nach dem Komma getrennt werden.

Wenn null oder mehrere Werte angegeben sind, vereinbaren der Client und der Server beim SSL-Handshake die Verwendung der sichersten Cipher Suites, die zu finden sind. Wenn keine kompatiblen Cipher Suites gefunden werden, schlägt die Verbindung fehl. Es ist nicht möglich, Prioritäten für Cipher Suites durch ihre Angabe in einer bestimmten Reihenfolge festzulegen.

ssl_clnt_keydb - SSL-Schlüsseldateipfad für abgehende SSL-Verbindungen auf dem Client (Konfigurationsparameter)

Dieser Konfigurationsparameter gibt die Schlüsseldatei an, die für die SSL-Verbindung auf der Clientseite verwendet werden soll.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
Null [beliebiger gültiger Pfad; GSK_MS_CERTIFICATE_STORE]

Mit diesem Parameter wird ein vollständig qualifizierter Dateipfad der Schlüsseldatei oder - nur unter Windows - das Schlüsselwort GSK_MS_CERTIFICATE_STORE angegeben. Dies zeigt die Verwendung des Zertifikatsspeichers von Microsoft Windows an.

Die SSL-Schlüsseldatei hat standardmäßig die Dateinamenerweiterung **.kbd** und dient zum Speichern des Unterzeichnerzertifikats aus dem persönlichen Zertifikat des Servers. Für ein selbst signiertes persönliches Zertifikat eines Servers ist das Unterzeichnerzertifikat der öffentliche Schlüssel. Für ein von einer Zertifizierungsstelle (CA) signiertes persönliches Zertifikat eines Servers ist das Unterzeichnerzertifikat das CA-Stammzertifikat. Auf die Schlüsseldatei wird vom Client zugegriffen, um das persönliche Zertifikat des Servers beim SSL-Handshake zu überprüfen.

Dieser Wert ist standardmäßig null. Abhängig vom Anwendungstyp sollten Sie den SSL-Schlüsseldateipfad für den Client durch den Konfigurationsparameter **ssl_clnt_keydb**, die Verbindungszeichenfolge **ssl_clnt_keydb** oder durch das Schlüsselwort **SSLClientKeystoredb** in den Dateien db2cli.ini und db2dsdriver.cfg für eine SSL-Verbindungsanforderung angeben. Wenn keine dieser Angaben vorhanden ist, schlägt die SSL-Verbindung fehl.

ssl_clnt_stash - SSL-Stashdateipfad für abgehende SSL-Verbindungen auf dem Client (Konfigurationsparameter)

Mit diesem Konfigurationsparameter wird der vollständig qualifizierte Dateipfad für die Stashdatei angegeben, die für SSL-Verbindungen auf der Clientseite verwendet werden soll.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
Null [beliebiger gültiger Pfad]

Die SSL-Stashdatei hat standardmäßig die Dateinamenerweiterung `.sth` und dient zum Speichern einer verschlüsselten Version des Kennworts für die Schlüsseldatenbank. Das Kennwort, das in der Stashdatei gespeichert ist, wird für den Zugriff auf die SSL-Schlüsseldatei bei einer SSL-Verbindungsanforderung verwendet.

Auf Windows-Plattformen ist **ssl_clnt_stash** nicht erforderlich, wenn für **ssl_clnt_keydb** das Schlüsselwort GSK_MS_CERTIFICATE_STORE angegeben ist.

Dieser Wert ist standardmäßig null. Abhängig von Ihrem Anwendungstyp können Sie den SSL-Stashdateipfad für den Client durch den Konfigurationsparameter **ssl_clnt_stash** des Datenbankmanagers, durch die Verbindungszeichenfolge **ssl_clnt_stash** oder durch das Schlüsselwort **ssl_clnt_stash** in der Datei `'db2cli.ini'` für eine SSL-Verbindungsanforderung angeben. Wenn keine dieser Angaben vorhanden ist, schlägt die SSL-Verbindung fehl.

ssl_svr_keydb - SSL-Schlüsseldateipfad für ankommende SSL-Verbindungen auf dem Server (Konfigurationsparameter)

Dieser Konfigurationsparameter gibt die Schlüsseldatei an, die für die SSL-Konfiguration auf der Serverseite verwendet werden soll.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
Null [beliebiger gültiger Pfad; GSK_MS_CERTIFICATE_STORE]

Mit diesem Parameter wird ein vollständig qualifizierter Dateipfad der Schlüsseldatei oder - nur unter Windows - das Schlüsselwort GSK_MS_CERTIFICATE_STORE angegeben. Dies zeigt die Verwendung des Zertifikatsspeichers von Microsoft Windows an.

Die SSL-Schlüsseldatei hat standardmäßig die Dateinamenerweiterung `.kdb` und dient zum Speichern von persönlichen Zertifikaten, Anforderungen von persönlichen Zertifikaten sowie von Unterzeichnerzertifikaten. Auf diese Schlüsseldatei wird während des Starts der Instanz zugegriffen und das persönliche Zertifikat des Servers wird an den Client gesendet, um die Serverauthentifizierung während des SSL-Handshakes auszuführen.

Der Wert ist standardmäßig null. Während des Instanzstarts müssen Sie definieren, ob die Registrierdatenbankvariable **DB2COMM** die Angabe `SSL` enthält. Ansonsten wird die Instanz ohne Unterstützung des SSL-Protokolls gestartet.

ssl_svr_label - Kennsatz in der Schlüsseldatei für ankommende SSL-Verbindungen auf dem Server (Konfigurationsparameter)

Mit diesem Konfigurationsparameter wird ein Kennsatz (Label) des persönlichen Zertifikats des Servers in der Schlüsseldatenbank angegeben.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert
Null

Der Wert ist standardmäßig null. Bei der Herstellung einer SSL-Verbindung wird das Serverzertifikat, das durch diesen Konfigurationsparameter angegeben wird, an den Client zur Serverauthentifizierung gesendet. Wenn der Wert null ist, wird das in der Schlüsseldatei definierte Standardzertifikat verwendet. Wenn kein Standardzertifikat vorhanden ist, schlägt die Verbindung fehl.

ssl_svr_stash - SSL-Stashdateipfad für ankommende SSL-Verbindungen auf dem Server (Konfigurationsparameter)

Mit diesem Konfigurationsparameter wird ein vollständig qualifizierter Dateipfad für die Stashdatei angegeben, die für die SSL-Konfiguration auf der Serverseite verwendet werden soll.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
Null [beliebiger gültiger Pfad]

Die SSL-Stashdatei hat standardmäßig die Dateinamenerweiterung `.sth` und dient zum Speichern einer verschlüsselten Version des Kennworts für die Schlüsseldatenbank. Das Kennwort, das in der Stashdatei gespeichert ist, wird für den Zugriff auf die SSL-Schlüsseldatei beim Start der Instanz verwendet.

Der Wert ist standardmäßig null. Während des Instanzstarts müssen Sie definieren, ob die Registrierdatenbankvariable **DB2COMM** die Angabe `SSL` enthält. Ansonsten wird die Instanz ohne Unterstützung des SSL-Protokolls gestartet.

Auf Windows-Plattformen ist **ssl_svr_stash** nicht erforderlich, wenn für **ssl_svr_keydb** das Schlüsselwort `GSK_MS_CERTIFICATE_STORE` angegeben ist.

start_stop_time - Zeitlimit für DB2START und DB2STOP

Dieser Parameter gibt die Zeit (in Minuten) an, innerhalb der alle Datenbankpartitionsserver auf einen Befehl zum Starten oder Stoppen des Datenbankmanagers (**START DBM** bzw. **STOP DBM**) reagieren müssen. Außerdem wird er als Zeitlimitwert für **ADD DBPARTITIONNUM**- und **DROP DBPARTITIONNUM**-Operationen verwendet.

Konfigurationstyp
Datenbankmanager

Gilt für
Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
`10 [1 - 1 440]`

Maßeinheit
Minuten

Member oder Knoten, die nicht innerhalb der angegebenen Zeit auf den Befehl **db2start** oder **db2stop** reagieren, werden in einer Instanz mit mehreren Membern/Knoten automatisch von **db2start** oder **db2stop** abgebrochen und bereinigt. Die Diagnosenachrichten werden in dem Pfad **diagpath** protokolliert, der in der Datenbankmanagerkonfiguration definiert ist oder mit dem zugehörigen Standardwert (z. B. `sqllib/db2dump/ $m` unter UNIX-Betriebssystemen).

Wenn eine **db2start**- oder **db2stop**-Operation in einer Mehrpartitionsdatenbank nicht innerhalb des Zeitraums ausgeführt wird, der durch den Konfigurationsparameter **start_stop_time** des Datenbankmanagers angegeben wurde, werden die Datenbankpartitionen automatisch abgebrochen und bereinigt, für die das zulässige Zeitlimit überschritten wurde. Bei Umgebungen mit vielen Datenbankpartitionen und einem geringen Wert für **start_stop_time** kann es zu diesem Verhalten kommen. Erhöhen Sie den Wert für **start_stop_time**, um diesem Verhalten entgegenzuwirken.

Beim Hinzufügen einer neuen Datenbankpartition mithilfe des Befehls **db2start**, **START DATABASE MANAGER** oder **ADD DBPARTITIONNUM** muss die Operation zum Hinzufügen der Datenbankpartition bestimmen, ob die einzelnen Datenbanken in der Instanz für dynamischen Speicher eingerichtet sind. Dies geschieht durch Kommunikation mit der Katalogpartition für jede einzelne Datenbank. Wenn der dynamische Speicher aktiviert ist, werden die Speicherpfaddefinitionen im Rahmen dieser Kommunikation abgerufen. Ähnlich gilt für den Fall, dass Tabellenbereiche für temporäre Tabellen zusammen mit den Datenbankpartitionen zu erstellen sind, dass die Operation möglicherweise mit einem anderen Datenbankpartitionsserver kommunizieren muss, um die Tabellenbereichsdefinitionen für die Datenbankpartitionen abzurufen, die sich auf diesem Server befinden. Diese Faktoren müssen bei der Festlegung des Werts für den Parameter **start_stop_time** berücksichtigt werden.

ssl_svcename - SSL-Servicename (Konfigurationsparameter)

Mit diesem Konfigurationsparameter wird der Name des Ports angegeben, über den ein Datenbankserver die Kommunikation von fernen Clientknoten unter Verwendung des SSL-Protokolls erwartet.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert
Null

Dieser Konfigurationsparameter enthält den Port, über den ein Datenbankserver die Kommunikation von fernen Clientknoten unter Verwendung des SSL-Protokolls erwartet. Der Servicename muss für die Verwendung durch den Datenbankmanager reserviert sein. Während des Instanzstarts müssen Sie definieren, ob die Registrierdatenbankvariable **DB2COMM** die Angabe SSL enthält. Andernfalls wird die Instanz ohne Unterstützung des SSL-Protokolls gestartet.

Wenn die Registrierdatenbankvariable **DB2COMM** beide Angaben, h. h. TCPIP und SSL enthält, darf der Port, der durch den Parameter **ssl_svcename** angegeben wird, nicht mit dem Port identisch sein, der durch den Parameter **svcename** angegeben wird. Andernfalls wird die Instanz ohne Unterstützung des SSL-Protokolls oder des TCP/IP-Protokolls gestartet.

Unter UNIX-Betriebssystemen befindet sich die Datei services in folgendem Verzeichnis: /etc/services

Die SSL-Portnummer des Datenbankservers (Nummer *n*) und der zugehörige Servicename müssen in der Servicedatei auf dem Datenbankclient definiert werden.

ssl_versions - Unterstützte SSL-Versionen auf dem Server (Konfigurationsparameter)

Mit diesem Konfigurationsparameter werden die Versionen von Secure Sockets Layer (SSL) und Transport Layer Security (TLS) angegeben, die der Server für ankommende Verbindungsanforderungen unterstützt.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
Null [TLSv1]

Wenn Sie den Parameter auf null oder den Wert TLSv1 setzen, aktiviert er die Unterstützung für TLS Version 1.0 (RFC2246) und TLS Version 1.1 (RFC4346).

Beim SSL-Handshake vereinbaren der Client und der Server die Verwendung der sichersten Version, die zu finden ist (d. h. entweder TLS Version 1.0 oder TLS Version 1.1). Wenn zwischen dem Client und dem Server keine kompatible Version vorhanden ist, schlägt der Verbindungsaufbau fehl. Wenn der Client TLS Version 1.0 und TLS Version 1.1 unterstützt, der Server jedoch nur TLS Version 1.0, wird TLS Version 1.0 verwendet.

svcename - TCP/IP-Servicename

Dieser Parameter enthält den Namen des TCP/IP-Ports, an dem ein Datenbankserver auf Datenübertragungen von fernen Clientknoten wartet. Dieser Name muss der für die Verwendung durch den Datenbankmanager reservierte Port sein.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert
Null

Damit Verbindungsanforderungen von einem Data Server Runtime Client mithilfe von TCP/IP empfangen werden können, muss der Datenbankserver an einem Port empfangsbereit sein, der diesem Server zugewiesen wurde. Der Systemadministrator für den Datenbankserver muss einen Port (Nummer *n*) reservieren und den zugeordneten TCP/IP-Servicenamen in der Servicedatei auf dem Server definieren.

Der Port (Nummer *n*) des Datenbankservers und sein TCP/IP-Servicename müssen in der Servicedatei auf dem Datenbankclient definiert werden.

Auf Linux- und UNIX-Systemen befindet sich die Servicedatei in folgendem Verzeichnis: /etc/services

Der Parameter **svcename** sollte auf die Portnummer oder den Servicenamen gesetzt werden, der dem Hauptverbindungsport zugeordnet ist, sodass der Datenbankserver nach dem Start ermitteln kann, an welchem Port er für eingehende Verbindungsanforderungen empfangsbereit sein muss.

sysadm_group - SYSADM-Gruppenname

Mit diesem Parameter wird der Gruppenname definiert, der die Berechtigung SYSADM für die Datenbankmanagerinstanz besitzt.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert
NULL

Die Berechtigung SYSADM (Systemadministrator) stellt die höchste Ebene der Berechtigungen auf der Instanzebene dar. Benutzer mit der Berechtigung SYSADM können innerhalb der Instanz bestimmte Dienstprogramme ausführen und bestimmte Datenbank- und Datenbankmanagerbefehle absetzen.

Die Berechtigung SYSADM wird durch die Sicherheitseinrichtungen festgelegt, die in einer bestimmten Betriebsumgebung verwendet werden.

- Für das Windows-Betriebssystem kann dieser Parameter auf die lokale Gruppe oder die Domänengruppe gesetzt werden. Gruppennamen müssen die Längenbegrenzungen einhalten, die in SQL- und XML-Begrenzungen angegeben sind. Die folgenden Benutzer verfügen über die Berechtigung SYSADM, wenn der Wert NULL für den Konfigurationsparameter **sysadm_group** des Datenbankmanagers angegeben wird:
  - Mitglieder der lokalen Administratorgruppe
  - Mitglieder der Gruppe 'Administratoren' auf dem Domänencontroller, wenn **DB2_GRP_LOOKUP** nicht auf den Wert DOMAIN gesetzt ist
  - Mitglieder der Gruppe DB2ADMNS, wenn die Funktion der erweiterten Sicherheit aktiviert ist. Die Position der Gruppe DB2ADMNS wurde bei der Installation festgelegt.
  - Das lokale Systemkonto (LocalSystem)
- Wenn bei Linux- und UNIX-Systemen der Wert NULL für diesen Parameter angegeben wird, gilt die Primärgruppe des Instanzeigners standardmäßig als SYSADM-Gruppe.

  Ist der Wert nicht NULL, kann als SYSADM-Gruppe jeder gültige UNIX-Gruppenname definiert werden.

Wenn Sie den Parameter auf seinen Standardwert (NULL) zurücksetzen wollen, führen Sie den Befehl **UPDATE DBM CFG USING SYSADM_GROUP NULL** aus. Das Schlüsselwort NULL muss in Großbuchstaben angegeben werden.

sysctrl_group - SYSCTRL-Gruppenname

Mit diesem Parameter wird der Gruppenname definiert, der die Berechtigung SYSCTRL (Systemsteuerung) besitzt. Die Berechtigung SYSCTRL umfasst Zugriffsrechte, die Operationen ermöglichen, die sich auf Systemressourcen auswirken, jedoch keinen direkten Zugriff auf Daten zulassen.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert
Null

Gruppennamen werden auf allen Plattformen akzeptiert, solange sie die in SQL- und XML-Begrenzungen angegebenen Längenbegrenzungen einhalten.

sysmaint_group - SYSMAINT-Gruppenname

Mit diesem Parameter wird der Gruppenname definiert, der die Berechtigung SYSMAINT (Systempflege) besitzt.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert
Null

Mit der Berechtigung SYSMAINT können Operationen zur Pflege aller Datenbanken, die zu einer Instanz gehören, durchgeführt werden, jedoch ohne direkten Zugriff auf Daten.

Gruppennamen werden auf allen Plattformen akzeptiert, solange sie die in SQL- und XML-Begrenzungen angegebenen Längenbegrenzungen einhalten.

Wenn Sie den Parameter auf seinen Standardwert (NULL) zurücksetzen wollen, führen Sie den Befehl **UPDATE DBM CFG USING SYSMAINT_GROUP NULL** aus. Das Schlüsselwort NULL muss in Großbuchstaben angegeben werden.

sysmon_group - Berechtigungsgruppenname für Systemmonitor

Mit diesem Parameter wird der Gruppenname definiert, der die Berechtigung SYSMON (Systemmonitor) besitzt.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert
`NULL`

Benutzer, die über die Berechtigung SYSMON auf der Instanzebene verfügen, können über den Datenbanksystemmonitor Momentaufnahmen von einer Datenbankmanagerinstanz oder den zugehörigen Datenbanken erstellen. Eine vollständige Liste der Befehle, die von der Berechtigung SYSMON unterstützt werden, finden Sie in Systemmonitorberechtigung (SYSMON).

Benutzer mit der Berechtigung SYSADM, SYSCTRL oder SYSMAINT haben automatisch die Möglichkeit, Momentaufnahmen über den Datenbanksystemmonitor zu erstellen und diese Befehle zu verwenden.

Gruppennamen werden auf allen Plattformen akzeptiert, solange sie die im Handbuch *Datenbankverwaltung - Konzepte und Konfiguration - Referenzinformationen* in „SQL- und XML-Begrenzungen" angegebenen Längenbegrenzungen einhalten.

Wenn Sie den Parameter auf seinen Standardwert (`NULL`) zurücksetzen wollen, führen Sie den Befehl **`UPDATE DBM CFG USING SYSMON_GROUP NULL`** aus. Das Schlüsselwort `NULL` muss in Großbuchstaben angegeben werden.

tm_database - Name für Transaktionsmanagerdatenbank

Mit diesem Parameter wird der Name der Transaktionsmanagerdatenbank (TMD) für jede DB2-Instanz angegeben.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
1ST_CONN [beliebiger gültiger Datenbankname]

Eine Transaktionsmanagerdatenbank kann eine der folgenden Datenbanken sein:

- Eine lokale DB2-Datenbank
- Eine ferne DB2-Datenbank, die sich nicht auf einem Host oder System AS/400 befindet
- Eine Datenbank von DB2 für OS/390 Version 5, wenn auf sie über TCP/IP zugegriffen und der Synchronisationspunktmanager (SPM) nicht verwendet wird

Es handelt sich dabei um eine Datenbank, die zu Protokoll- und Koordinierungsfunktionen verwendet wird und die zur Recovery unbestätigter Transaktionen dient.

Dieser Parameter kann auf den Wert **1ST_CONN** gesetzt werden, wodurch festgelegt wird, dass die Transaktionsmanagerdatenbank die erste Datenbank ist, zu der ein Benutzer eine Verbindung herstellt.

Empfehlung: Zur Vereinfachung der Verwaltung und des Betriebs können Sie einige Datenbanken in verschiedenen Instanzen erstellen und diese Datenbanken ausschließlich als Transaktionsmanagerdatenbanken verwenden.

tp_mon_name - Name des Transaktionsprozessormonitors

Mit diesem Parameter wird der Name des verwendeten Monitors für das Transaktionsprogramm (TP) angegeben.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert
Kein Standardwert

Gültige Werte

- CICS
- MQ
- CB
- SF
- TUXEDO
- TOPEND
- Kein Eintrag oder ein weiterer Wert (unter UNIX und Windows; keine weiteren gültigen Werte unter Solaris oder SINIX)

- Sind Anwendungen in einer WebSphere Enterprise Server Edition CICS-Umgebung aktiv, muss für den Parameter „CICS" angegeben werden.
- Sind Anwendungen in einer WebSphere Enterprise Server Edition Component Broker-Umgebung aktiv, muss für den Parameter „CB" angegeben werden.
- Sind Anwendungen in einer IBM MQSeries-Umgebung aktiv, muss für den Parameter „MQ" angegeben werden.

- Sind Anwendungen in einer BEA Tuxedo-Umgebung aktiv, muss für den Parameter „TUXEDO" angegeben werden.
- Sind Anwendungen in einer IBM San Francisco-Umgebung aktiv, muss für den Parameter „SF" angegeben werden.

Benutzer von **IBM WebSphere EJB** und **Microsoft Transaction Server** müssen für diesen Parameter keinen Wert konfigurieren.

Wenn keines der oben genannten Produkte verwendet wird, sollte dieser Parameter nicht konfiguriert, sondern ohne Angabe eines Werts belassen werden.

In vorangehenden Versionen von IBM DB2 unter Windows enthielt dieser Parameter den Pfad und den Namen der DLL, in der die Funktionen *ax_reg* und *ax_unreg* des XA-Transaktionsmanagers gespeichert waren. Dieses Format wird noch immer unterstützt. Wenn der Wert für diesen Parameter mit keinem der oben genannten TP-Monitornamen übereinstimmt, wird angenommen, dass der angegebene Wert der Name einer Bibliothek ist, die die Funktionen *ax_reg* und *ax_unreg* enthält. Dies gilt für UNIX- und Windows-Umgebungen.

Benutzer von TXSeries CICS: In vorangegangenen Versionen dieses Produkts unter Windows war es erforderlich, diesen Parameter als „libEncServer:C" oder „libEncServer:E" zu konfigurieren. Dies wird noch immer unterstützt, ist aber nicht mehr erforderlich. Wenn der Parameter als „CICS" konfiguriert wird, ist dies ausreichend.

Benutzer von MQSeries: In vorangegangenen Versionen dieses Produkts unter Windows war es erforderlich, diesen Parameter als „mqmax" zu konfigurieren. Dies wird noch immer unterstützt, ist aber nicht mehr erforderlich. Wenn der Parameter als „MQ" konfiguriert wird, ist dies ausreichend.

Benutzer von Component Broker: In vorangegangenen Versionen dieses Produkts unter Windows war es erforderlich, diesen Parameter als „somtrx1i" zu konfigurieren. Dies wird noch immer unterstützt, ist aber nicht mehr erforderlich. Wenn der Parameter als „CB" konfiguriert wird, ist dies ausreichend.

Benutzer von San Francisco: In vorangegangenen Versionen dieses Produkts unter Windows war es erforderlich, diesen Parameter als „ibmsfDB2" zu konfigurieren. Dies wird noch immer unterstützt, ist aber nicht mehr erforderlich. Wenn der Parameter als „SF" konfiguriert wird, ist dies ausreichend.

Die maximale Länge der Zeichenfolge, die für diesen Parameter angegeben werden kann, beträgt 19 Zeichen.

Diese Informationen können auch in der Zeichenfolge XA OPEN von IBM DB2 Version 9.1 konfiguriert werden. Wenn mehrere TP-Monitore in einer einzigen DB2-Instanz verwendet werden, muss diese Funktion verwendet werden.

trust_allclnts - Alle Clients akzeptieren

Dieser Parameter und der Parameter **trust_clntauth** werden verwendet, um festzustellen, wo Benutzerberechtigungen für die Datenbankumgebung überprüft werden.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
YES [NO, YES, DRDAONLY]

Dieser Parameter ist nur aktiviert, wenn der Parameter **authentication** auf CLIENT gesetzt ist.

Durch Übernehmen des Standardwerts YES für diesen Parameter werden alle Clients als gesicherte Clients akzeptiert. Dies bedeutet, der Server geht davon aus, dass auf jedem Client eine Sicherheitsstufe vorhanden ist und auf jedem Client eine Gültigkeitsprüfung für Benutzer ausgeführt werden kann.

Dieser Parameter kann nur auf NO gesetzt werden, wenn der Parameter **authentication** auf CLIENT gesetzt ist. Ist dieser Parameter auf NO gesetzt, müssen die ungesicherten Clients beim Herstellen einer Verbindung zum Server eine Benutzer-ID mit Kennwort angeben. Ungesicherte Clients sind Betriebssystemplattformen, die nicht über ein Sicherheitssubsystem zur Gültigkeitsprüfung für Benutzer verfügen.

Das Setzen dieses Parameters auf DRDAONLY schützt vor allen Clients mit Ausnahme von Clients von DB2 für OS/390 und z/OS, DB2 für VM und VSE sowie DB2 für OS/400. Nur diese Clients dürfen die Authentifizierung auf der Clientseite ausführen. Alle anderen Clients müssen eine Benutzer-ID und ein Kennwort bereitstellen, wobei die Authentifizierung vom Server durchgeführt wird.

Wenn **trust_allclnts** auf DRDAONLY gesetzt ist, wird mit dem Parameter **trust_clntauth** ermittelt, wo die Authentifizierung der Clients erfolgt. Wenn **trust_clntauth** auf CLIENT gesetzt ist, findet die Authentifizierung auf dem Client statt. Wenn **trust_clntauth** auf SERVER gesetzt ist, erfolgt die Authentifizierung auf dem Client, sofern kein Kennwort angegeben ist, und auf dem Server, sofern ein Kennwort angegeben ist.

trust_clntauth - Authentifizierung gesicherter Clients

Dieser Parameter gibt an, ob eine Authentifizierung gesicherter Clients auf dem Server oder auf dem Client erfolgt, wenn der Client eine Benutzer-ID mit Kennwort für eine Verbindung angibt.

Dieser Parameter (und **trust_allclnts**) ist nur aktiviert, wenn der Parameter **authentication** auf CLIENT gesetzt ist. Wird keine Benutzer-ID mit Kennwort angegeben, geht das System davon aus, dass die Authentifizierung des Benutzers vom Client ausgeführt wurde, d. h., auf dem Server erfolgt keine weitere Überprüfung.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
`CLIENT [CLIENT, SERVER]`

Wenn dieser Parameter auf `CLIENT` (die Standardeinstellung) gesetzt ist, kann der gesicherte Client eine Verbindung herstellen, ohne eine Benutzer-ID mit Kennwort anzugeben, und es wird angenommen, dass die Authentifizierung des Benutzers bereits vom Betriebssystem vorgenommen wurde. Ist der Parameter auf `SERVER` gesetzt, werden Benutzer-ID und Kennwort auf dem Server überprüft.

Wenn die Anwendungsprogrammierschnittstelle db2CfgSet verwendet wird, um den Konfigurationsparameter des Datenbankmanagers zu definieren, ist 0 der numerische Wert für CLIENT. Der numerische Wert für SERVER ist 1.

util_impact_lim - Richtlinie für Instanzauslastungswirkung

Dieser Parameter ermöglicht dem Datenbankadministrator (DBA), die Leistungseinbußen für eine Auslastung durch ein gedrosseltes Dienstprogramm zu begrenzen.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen Clients
- Datenbankserver mit lokalen und fernen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
`10 [1 - 100 ]`

Maßeinheit
Prozentsatz der zulässigen Wirkung auf die Auslastung

Wenn die Leistungseinbußen begrenzt sind, kann der Datenbankadministrator während kritischer Auslastungszeiten des Produktionssystems Onlinedienstprogramme ausführen und so sicher sein, dass die Auswirkung auf die Produktionsleistung innerhalb annehmbarer Grenzen bleibt.

Zum Beispiel kann ein Datenbankadministrator, der den Parameter **`util_impact_lim`** (Auslastungswirkung) auf den Wert `10` setzt, davon ausgehen, dass ein gedrosselter BACKUP-Aufruf die Auslastung mit nicht mehr als 10 Prozent beansprucht.

Wenn der Parameter **`util_impact_lim`** auf den Wert `100` gesetzt ist, findet keine Drosselung von Dienstprogrammaufrufen statt. In diesem Fall können die Dienstprogramme die Auslastung in willkürlichem (und unerwünschtem) Ausmaß beanspruchen. Wenn der Parameter **`util_impact_lim`** auf einen Wert unter `100` gesetzt ist, ist es möglich, Dienstprogramme im gedrosselten Modus aufzurufen. Zur Ausführung im gedrosselten Modus muss ein Dienstprogramm außerdem mit einer Priorität ungleich Null aufgerufen werden.

Empfehlung: Für die meisten Benutzer ist es wahrscheinlich vorteilhaft, wenn der Parameter **util_impact_lim** auf einen niedrigen Wert (z. B. zwischen 1 und 10) gesetzt wird.

Ein gedrosseltes Dienstprogramm benötigt in der Regel mehr Ausführungszeit als ein ungedrosseltes Dienstprogramm. Wenn Sie feststellen, dass ein Dienstprogramm extrem viel Zeit benötigt, erhöhen Sie den Wert des Parameters **util_impact_lim** oder inaktivieren Sie die Drosselung völlig, indem Sie den Parameter **util_impact_lim** auf den Wert 100 setzen.

wlm_dispatcher - Workload-Management-Dispatcher

Dieser Parameter aktiviert (YES) oder inaktiviert (NO) den DB2 Workload Management-Dispatcher (DB2 WLM-Dispatcher). Ein aktivierter WLM-Dispatcher ermöglicht standardmäßig die Einstellung der CPU-Begrenzungen.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
NO [NO; YES]

Beim Aktualisieren des DB2-Datenbankmanagers wird als Wert für den Konfigurationsparameter **wlm_dispatcher** des Datenbankmanagers der Wert NO eingestellt.

Der WLM-Dispatcher stellt Funktionen für die CPU-Zeitplanung auf Serviceklassenebene im DB2-Datenbankmanager durch Zuteilen von CPU-Ressourcen in Form von CPU-Anteilen und/oder CPU-Begrenzungen zur Verfügung.

Wenn der WLM-Dispatcher aktiviert ist, werden alle Arbeitsoperationen in Benutzer- und Wartungsserviceklassen der Steuerung durch den Dispatcher unterstellt. Ist der Dispatcher aktiviert, werden die Einstellungen für CPU-Begrenzungen standardmäßig durch den Dispatcher erzwungen. Damit die Zuteilung von CPU-Ressourcen in Form von CPU-Anteilen verwendet werden kann, muss der Konfigurationsparameter **wlm_disp_cpu_shares** des Datenbankmanagers aktiviert sein.

Wenn der Konfigurationsparameter **wlm_dispatcher** auf YES gesetzt ist, gelten die folgenden Bedingungen:

- Wenn eine Serviceklasse über eine Prioritätseinstellung verfügt, die nicht der Standardeinstellung entspricht, wird im Zeitpunkt der Datenbankaktivierung eine Warnung in das DB2-Diagnoseprotokoll (db2diag) und in das Protokoll mit Benachrichtigungen für die Systemverwaltung geschrieben.
- Bei jedem Versuch, für die Agentenpriorität in einer Serviceklasse einen anderen als den Standardwert festzulegen, wird eine Warnung an die Anwendung zurückgegeben, von der die Anweisung zum Erstellen oder Ändern der Serviceklasse ausgeführt wurde.

wlm_disp_concur - Gemeinsamer Threadzugriff für Workload-Manager-Dispatcher ()

Dieser Parameter gibt an, wie der DB2 Workload Manager-Dispatcher (WLM-Dispatcher) die Einstellung für den gemeinsamen Threadzugriff festlegt. Sie können auch manuell einen festen Wert für den gemeinsamen Threadzugriff festlegen.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
COMPUTED [COMPUTED; *manuell_festgelegter_wert*]

Beim Aktualisieren des DB2-Datenbankmanagers hat der Konfigurationsparameter **wlm_disp_concur** den Wert COMPUTED.

COMPUTED
Der DB2-Datenbankmanager berechnet einen festen Wert für den gemeinsamen Threadzugriff auf der Basis des Vierfachen der Anzahl logischer CPUs, die für den DB2-Datenbankmanager verfügbar sind.

manuell_festgelegter_wert
Sie können manuell einen festen Wert für den gemeinsamen Threadzugriff im Bereich von 1 bis 32767 festlegen. Der optimale Wert ist abhängig von der verwendeten Hardware und Betriebssystemversion und liegt im Allgemeinen im Bereich zwischen dem Zwei- und Vierfachen der Anzahl logischer CPUs auf dem Host oder in der logischen Partition (LPAR).

Maßeinheit
Anzahl der gleichzeitig ausgeführten Threads

Die Einstellung für diesen Konfigurationsparameter des Datenbankmanagers steuert, für wie viele Threads der WLM-Dispatcher die gleichzeitige (parallele) Zuteilung zu den Ausführungswarteschlangen des Betriebssystems zulässt. Als Wert wird ein kleines Vielfaches der Anzahl logischer CPUs angegeben, die für den DB2-Datenbankmanager verfügbar sind. In der Regel kann für diesen Wert das Vierfache der Anzahl verfügbarer logischer CPUs angegeben werden, um mögliche Latenzzeiten der Zeitplanung zu berücksichtigen, die beim Umschalten von Threads zwischen dem aktiven und inaktiven Status entstehen können. Ein optimaler Wert ist gerade groß genug, um sicherzustellen, dass genügend Threads für den DB2-Datenbankmanager verfügbar sind, um die CPUs auf dem Host oder in der LPAR in vollem Umfang zu nutzen. Dieser optimale Wert sorgt für maximale Effizienz und gibt dem DB2-WLM-Dispatcher maximale Kontrolle über die CPU-Zuordnung.

wlm_disp_cpu_shares - CPU-Anteile des Workload-Manager-Dispatchers ()

Dieser Parameter aktiviert (Einstellung YES) oder inaktiviert (Einstellung NO) die Steuerung der CPU-Anteile durch den DB2 Workload Manager-Dispatcher (DB2 WLM-Dispatcher). Standardmäßig steuert ein aktivierter WLM-Dispatcher nur CPU-Begrenzungen.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
NO [NO; YES]

Beim Aktualisieren des DB2-Datenbankmanagers hat der Konfigurationsparameter **wlm_disp_cpu_shares** des Datenbankmanagers den Wert NO.

Wenn der Wert des Konfigurationsparameters **wlm_dispatcher** des Datenbankmanagers auf YES und der Wert des Konfigurationsparameters **wlm_disp_cpu_shares** des Datenbankmanagers auf NO gesetzt ist, kann der WLM-Dispatcher bei der Verwaltung von Serviceklassen nur CPU-Begrenzungen anwenden.

Ist der Wert des Konfigurationsparameters **wlm_dispatcher** des Datenbankmanagers auf YES gesetzt und der Wert des Konfigurationsparameters **wlm_disp_cpu_shares** des Datenbankmanagers auf YES, kann der WLM-Dispatcher bei der Verwaltung von Serviceklassen sowohl CPU-Begrenzungen als auch CPU-Anteile anwenden. Standardmäßig werden jeder Serviceklasse 1.000 feste CPU-Anteile zugeordnet, um eine gleichmäßige Aufteilung der CPU-Ressourcen zu gewährleisten.

Tabelle 136. Zusammenfassung der erforderlichen Einstellungen der Datenbankmanager-Konfigurationsparameter für die Serviceklassenverwaltung durch den DB2-WLM-Dispatcher

Serviceklassenverwaltung	Einstellung für wlm_dispatcher	Einstellung für wlm_disp_cpu_shares
Keine	NO	NO
CPU-Begrenzungen	YES	NO
CPU-Begrenzungen + CPU-Anteile	YES	YES

wlm_disp_min_util - Minimale CPU-Auslastung des Workload-Manager-Dispatchers (

Dieser Parameter gibt die minimale CPU-Auslastung an, die erforderlich ist, um eine Serviceklasse in die vom DB2-WLM verwaltete gemeinsame Nutzung von CPU-Ressourcen einzubinden.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Datenbankserver mit mehreren Membern mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
5 [0 bis 100]

Bei einen Upgrade des DB2-Datenbankmanagers weist der Konfigurationsparameter **wlm_disp_min_util** des Datenbankmanagers den Wert 5 auf.

Maßeinheit
Prozent

Ein Beispiel für die Verwendung dieses Datenbankmanager-Konfigurationsparameters: Angenommen es gibt drei Serviceklassen (A, B und C), die jeweils über 1.000 Anteile der CPU-Ressourcen verfügen. In diesem Beispiel wird stets das gleiche Ergebnis erzielt, unabhängig davon, ob es sich um feste oder veränderbare CPU-Anteile handelt. Die CPU-Auslastung der Serviceklassen A und B ist jeweils größer oder gleich dem Wert (8%), der für den Konfigurationsparameter **wlm_disp_min_util** angegeben ist. Die CPU-Auslastung der Serviceklasse C beträgt 3%, d. h. sie ist kleiner als der Wert (8%), der für den Konfigurationsparameter **wlm_disp_min_util** angegeben ist. In den Berechnungen für die CPU-Anteile wird davon ausgegangen, dass in der Klasse C keine Arbeitseinheiten zur Ausführung vorliegen. Demzufolge werden die CPU-Ressourcen gleichmäßig auf die Serviceklassen A und B verteilt, d. h. jeder Klasse werden 50% der CPU-Ressourcen zugeteilt. Sobald in der Serviceklasse Arbeitselemente verarbeitet werden, die eine CPU-Auslastung größer oder gleich dem im Konfigurationsparameter **wlm_disp_min_util** angegebenen Wert von 8% verursachen, werden alle drei Serviceklassen als gleichrangige Kandidaten für die Nutzung der CPU-Ressourcen angesehen, d. h. jeder Klasse werden 33,3% der CPU-Ressourcen zugeteilt.

In Datenbankumgebungen mit mehreren Membern wird die Summe der CPU-Anteile aller Member auf einem Host oder in einer logischen Partition (LPAR) mit dem Wert des Konfigurationsparameters **wlm_disp_min_util** verglichen, um festzulegen, ob der Host bzw. die LPAR in die gemeinsame Nutzung der von WLM verwalteten CPU-Ressourcen einbezogen wird.

Konfigurationsparameter der Datenbank

alt_collate - Alternative Sortierfolge

Mit diesem Parameter wird die Sortierfolge angegeben, die für Unicode-Tabellen in einer Nicht-Unicode-Datenbank zu verwenden ist.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients

- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
Null [IDENTITY_16BIT]

Unicode-Tabellen und -Routinen können erst in einer Nicht-Unicode-Datenbank erstellt werden, wenn dieser Parameter definiert wurde. Wenn dieser Parameter einmal definiert ist, kann er nicht mehr geändert oder zurückgesetzt werden.

Dieser Parameter kann nicht für Unicode-Datenbank definiert werden.

app_ctl_heap_sz - Zwischenspeichergröße für Anwendungssteuerung

Dieser Parameter ist in Version 9.5 veraltet, wird jedoch von Datenservern und Clients mit Versionen vor Version 9.5 weiterhin verwendet. Jeder für diesen Konfigurationsparameter angegebene Wert wird vom Datenbankmanager in DB2 Version 9.5 ignoriert. In Version 9.5 wurde er durch den Konfigurationsparameter **appl_memory** ersetzt.

Anmerkung: Die nachfolgenden Informationen gelten nur für Datenserver und Clients einer Version vor Version 9.5.

Bei partitionierten Datenbanken und nicht partitionierten Datenbanken mit aktivierter partitionsinterner Parallelität (**intra_parallel**=ON) gibt dieser Parameter die durchschnittliche Größe des gemeinsam genutzten Speicherbereichs an, der einer Anwendung zugeordnet ist. Bei nicht partitionierten Datenbanken, bei denen die partitionsinterne Parallelität nicht aktiviert ist (**intra_parallel**=OFF), ist dies der maximale private Speicher, der dem Zwischenspeicher zugeordnet wird. Pro Verbindung einer Datenbankpartition gibt es einen Zwischenspeicher für Anwendungssteuerung.

Konfigurationstyp
Datenbank

Parametertyp
Konfigurierbar

Standardwert [Bereich]

Datenbankserver mit lokalen und fernen Clients

- 128 [1 - 64 000] wenn **intra_parallel** nicht aktiviert wurde
- 512 [1 - 64 000] wenn **intra_parallel** aktiviert wurde

Datenbankserver mit lokalen Clients

- 64 [1 - 64 000] (für Nicht-UNIX-Plattformen) wenn **intra_parallel** nicht aktiviert wurde
- 512 [1 - 64 000] (für Nicht-UNIX-Plattformen) wenn **intra_parallel** aktiviert wurde
- 128 [1 - 64 000] (für Linux- und UNIX-Plattformen) wenn **intra_parallel** nicht aktiviert wurde
- 512 [1 - 64 000] (für Linux- und UNIX-Plattformen) wenn **intra_parallel** aktiviert wurde

Partitionierter Datenbankserver mit lokalen und fernen Clients
512 [1 - 64 000]

Maßeinheit
Seiten (4 KB)

Zuordnung
Beim Starten einer Anwendung

Freigabe
Beim Beenden einer Anwendung

Der Zwischenspeicher für Anwendungssteuerung wird in erster Linie für die gemeinsame Nutzung von Informationen durch Agenten benötigt, die für dieselbe Anforderung arbeiten. Die Auslastung dieses Zwischenspeichers ist bei nicht partitionierten Datenbanken minimal, wenn Abfragen mit einem Parallelitätsgrad gleich 1 ausgeführt werden.

Dieser Zwischenspeicher wird auch zum Speichern von Deskriptorinformationen für deklarierte temporäre Tabellen verwendet. Die Deskriptorinformationen für alle deklarierten temporären Tabellen, die nicht explizit gelöscht wurden, werden in diesem Zwischenspeicher aufbewahrt und können erst nach dem Löschen der deklarierten temporären Tabelle gelöscht werden.

Empfehlung: Verwenden Sie zunächst den Standardwert. Sie müssen den Wert eventuell erhöhen, wenn Sie komplexe Anwendungen ausführen oder ein System mit zahlreichen Datenbankpartitionen bzw. deklarierte temporäre Tabellen verwenden. Die erforderliche Speicherkapazität erhöht sich mit der Anzahl deklarierter temporärer Tabellen, die gleichzeitig aktiv sind. Der Tabellendeskriptor einer deklarierten temporären Tabelle mit vielen Spalten ist größer als jener einer Tabelle mit wenigen Spalten. Daher erhöht eine große Anzahl Spalten in den deklarierten temporären Tabellen einer Anwendung auch den Bedarf an Zwischenspeicher zur Anwendungssteuerung.

appgroup_mem_sz - Maximale Speichergröße für Anwendungsgruppe

Dieser Parameter ist in Version 9.5 veraltet, wird jedoch von Datenservern und Clients mit Versionen vor Version 9.5 weiterhin verwendet. Jeder für diesen Konfigurationsparameter angegebene Wert wird vom Datenbankmanager in DB2 Version 9.5 ignoriert. In Version 9.5 wurde er durch den Konfigurationsparameter **appl_memory** ersetzt.

Anmerkung: Die nachfolgenden Informationen gelten nur für Datenserver und Clients einer Version vor Version 9.5.

Dieser Parameter legt die Größe des von der Anwendungsgruppe gemeinsam genutzten Speichersegments fest.

Konfigurationstyp
Datenbank

Parametertyp
Konfigurierbar

Standardwert [Bereich]

UNIX-Datenbankserver mit lokalen Clients (nicht 32-Bit-HP-UX)
20 000 [1 - 1 000 000]

32-Bit-HP-UX

- Datenbankserver mit lokalen Clients
- Datenbankserver mit lokalen und fernen Clients
- Partitionierter Datenbankserver mit lokalen und fernen Clients

10 000 [1 - 1 000 000]

Windows-Datenbankserver mit lokalen Clients
10 000 [1 - 1 000 000]

Datenbankserver mit lokalen und fernen Clients (nicht 32-Bit-HP-UX)
30 000 [1 - 1 000 000]

Partitionierter Datenbankserver mit lokalen und fernen Clients (nicht 32-Bit-HP-UX)
40 000 [1 - 1 000 000]

Maßeinheit
Seiten (4 KB)

Informationen, die den für dieselbe Anwendung arbeitenden Agenten gemeinsam zur Verfügung stehen müssen, werden im gemeinsam genutzten Speichersegment der Anwendungsgruppe gespeichert.

In einer partitionierten Datenbank oder in einer nicht partitionierten Datenbank mit aktivierter partitionsinterner Parallelität oder aktiviertem Konzentrator nutzen mehrere Anwendungen eine Anwendungsgruppe gemeinsam. Der Anwendungsgruppe wird ein gemeinsam genutztes Speichersegment zugeordnet. Innerhalb des gemeinsam genutzten Speichersegments der Anwendungsgruppe ist für jede Anwendung ein eigener Zwischenspeicher für die Anwendungssteuerung vorhanden, und alle Anwendungen der Anwendungsgruppe nutzen gemeinsam einen Zwischenspeicher.

Die Anzahl Anwendungen in einer Anwendungsgruppe wird wie folgt berechnet:

```
appgroup_mem_sz / app_ctl_heap_sz
```

Die Größe des von der Anwendungsgruppe gemeinsam genutzten Zwischenspeichers wird wie folgt berechnet:

```
appgroup_mem_sz * groupheap_ratio / 100
```

Die Größe des Zwischenspeichers für die Anwendungssteuerung wird für jede Anwendung wie folgt berechnet:

```
app_ctl_heap_sz * (100 - groupheap_ratio) / 100
```

Empfehlung: Behalten Sie den Standardwert für diesen Parameter bei, solange Sie keine Leistungsprobleme feststellen.

appl_memory - Anwendungsspeicher (Konfigurationsparameter)

Mit diesem Parameter können Datenbankadministratoren (DBAs) und unabhängige Softwareanbieter (ISVs) die maximale Menge an Anwendungsspeicher steuern, die Serviceanwendungsanforderungen von DB2-Datenbankagenten zugeordnet wird.

Standardmäßig ist der Wert dieses Parameters auf `AUTOMATIC` gesetzt, was bedeutet, dass alle Anforderungen für Anwendungsspeicher zulässig sind, so lange die ge-

samte von der Datenbankpartition zugeordnete Speicherkapazität sich innerhalb der Grenzwerte des Parameters **instance_memory** bewegt.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp

- Online konfigurierbar
- Nach Member in einer DB2 pureScale-Umgebung konfigurierbar

Standardwert [Bereich]
Automatic [128 - 4 294 967 295]

Maßeinheit
Seiten (4 KB)

Zuordnung
Während der Aktivierung der Datenbank

Freigabe
Während der Inaktivierung der Datenbank

Anmerkung: Wenn **appl_memory** auf AUTOMATIC gesetzt wurde, ist die anfängliche Anwendungsspeicherzuordnung bei Aktivierung der Datenbank nur gering, sie nimmt jedoch nach Bedarf zu (oder ab). Diese Änderung wird im Speicher angewendet. Der Wert des Parameters **appl_memory** ändert sich auf der Platte nicht, wie sich durch den Befehl **db2 get db cfg show detail** anzeigen lässt. Bei der nächsten Aktivierung wird der Wert neu berechnet. Wenn der Parameter **appl_memory** auf einen bestimmten Wert gesetzt wurde, wird die erforderliche Speicherkapazität von Anfang an bei Aktivierung der Datenbank zugeordnet, und die Größe des Anwendungsspeichers ändert sich nicht. Wenn die anfängliche Anwendungsspeicherkapazität vom Betriebssystem nicht zugeordnet werden kann oder den Grenzwert des Parameters **instance_memory** überschreitet, schlägt die Aktivierung der Datenbank mit dem folgenden SQL-Fehler fehl: SQL1084C Gemeinsam genutzte Speichersegmente können nicht zugeordnet werden.

applheapsz - Zwischenspeichergröße für Anwendungen

Der Konfigurationsparameter **applheapsz** bezieht sich auf die gesamte Anwendungsspeichergröße, die von der gesamten Anwendung verwendet werden kann.

In DB2-Versionen vor Version 9.5 verweist der Datenbankkonfigurationsparameter **applheapsz** auf die Größe des Anwendungsspeichers, die die einzelnen für die Anwendung aktiven Datenbankagenten in Anspruch nehmen durften.

Ab Version 9.5 hat dieser Datenbankkonfigurationsparameter den Standardwert AUTOMATIC, was bedeutet, dass er nach Bedarf erhöht wird, bis entweder der Grenzwert des Parameters **appl_memory** oder des Parameters **instance_memory** erreicht ist. Für Umgebungen mit partitionierten Datenbanken und für Konzentrator- oder SMP-Konfigurationen bedeutet dies, dass der in früheren Releases verwendete Wert für den Parameter **applheapsz** bei ähnlichen Auslastungen eventuell erhöht werden muss, sofern nicht die Einstellung AUTOMATIC verwendet wird.

Konfigurationstyp
Datenbank

Parametertyp

- Online konfigurierbar
- Nach Member in einer DB2 pureScale-Umgebung konfigurierbar

Standardwert [Bereich]
Automatic [16 - 60 000]

Anmerkung: Der Standardwert unterliegt nach der ursprünglichen Datenbankerstellung Änderungen durch den DB2 Configuration Advisor.

Maßeinheit
Seiten (4 KB)

Zuordnung
Wenn eine Anwendung einer Datenbank zugeordnet wird oder zu ihr eine Verbindung herstellt.

Freigabe
Wenn die Zuordnung der Anwendung zur Datenbank aufgehoben oder die Verbindung der Anwendung zur Datenbank unterbrochen wird.

Anmerkung: Dieser Parameter definiert die maximale Größe des Anwendungszwischenspeichers. Pro Datenbankanwendung wird ein Anwendungszwischenspeicher zugeordnet, wenn die Anwendung das erste Mal mit der Datenbank verbunden wird. Der Zwischenspeicher wird von allen Datenbankagenten, die für die Anwendung aktiv sind, gemeinsam genutzt. (In früheren Releases ordneten sich die Datenbankagenten jeweils ihre eigenen Anwendungszwischenspeicher zu). Der Anwendungszwischenspeicher ordnet die Speicherkapazität, die für die Verarbeitung der Anwendung notwenig ist, gemäß dem Grenzwert zu, der über den Parameter definiert ist. Wenn die Einstellung `AUTOMATIC` gewählt wurde, wird der Anwendungszwischenspeicher bei Bedarf vergrößert, bis entweder der Grenzwert **`appl_memory`** für die Datenbank oder der Grenzwert **`instance_memory`** für die Datenbankpartition erreicht ist. Wenn die Verbindung der Anwendung zur Datenbank unterbrochen wird, wird der gesamte Anwendungszwischenspeicher freigegeben.

Der online geänderte Wert wird beim Grenzwert einer Anwendungsverbindung wirksam, d. h. wenn der Wert dynamisch geändert wurde, verwenden die zu dem Zeitpunkt verbundenen Anwendungen noch den alten Wert, während alle neu verbundenen Anwendungen den neuen Wert benutzen.

archretrydelay - Wiederholungsintervall für Archivierung bei Fehler

Mit diesem Parameter wird die Anzahl von Sekunden angegeben, die nach einem fehlgeschlagenen Archivierungsversuch abzuwarten ist, bevor die Archivierung der Protokolldatei erneut versucht wird.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
20 [0 - 65 535]

Nachfolgende Wiederholungen finden nur statt, wenn der Datenbankkonfigurationsparameter **numarchretry** mindestens auf den Wert 1 gesetzt ist.

auto_del_rec_obj - Automatisches Löschen von Recovery-Objekten (Konfigurationsparameter)

Dieser Parameter gibt an, ob Datenbankprotokolldateien, Backup-Images und Ladekopieimages gelöscht werden sollen, wenn ihr zugeordneter Eintrag in der Datei des Recoveryprotokolls bereinigt wird.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
OFF [ON; OFF]

Sie können die Einträge in der Datei des Recoveryprotokolls mithilfe des Befehls **PRUNE HISTORY** oder der API db2Prune bereinigen. Sie können auch den IBM Data Server-Datenbankmanager so konfigurieren, dass die Datei des Recoveryprotokolls nach jedem Datenbank-Backup automatisch bereinigt wird. Wenn Sie den Datenbankkonfigurationsparameter **auto_del_rec_obj** auf ON setzen, löscht der Datenbankmanager beim Bereinigen der Protokolldatei auch die entsprechenden physischen Protokolldateien, Backup-Images sowie Ladekopieimages. Der Datenbankmanager kann nur Recovery-Objekte wie Datenbankprotokolle, Backup-Images und Ladekopieimages löschen, wenn es sich bei Ihrem Speichermedium um eine Platte handelt oder wenn Sie einen Speichermanager wie Tivoli Storage Manager verwenden. Wenn der Parameter **logarchmeth1** auf den Wert LOGRETAIN gesetzt ist und der Befehl **ARCHIVE LOG** ausgegeben wird, werden die Protokolldateien nicht durch die Bereinigung gelöscht, auch wenn in der Protokolldatei Einträge vorhanden sind und **auto_del_rec_obj** auf ON gesetzt ist.

auto_maint - Automatische Verwaltung

Dieser Parameter ist das übergeordnete Element für alle anderen Datenbankkonfigurationsparameter der automatischen Verwaltung (**auto_db_backup**, **auto_tbl_maint**, **auto_runstats**, **auto_stats_prof**, **auto_stmt_stats**, **auto_stats_views**, **auto_prof_upd**, **auto_reorg** und **auto_sampling**).

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
ON [ON; OFF]

Wenn dieser Parameter inaktiviert wird, werden alle untergeordneten Parameter ebenfalls inaktiviert, jedoch ändern sich ihre Einstellungen, die in der Datenbankkonfigurationsdatei aufgezeichnet sind, nicht. Wenn dieser übergeordnete Parameter aktiviert wird, werden die aufgezeichneten Werte für die untergeordneten Parameter wirksam. Auf diese Weise kann die automatische Verwaltung global aktiviert bzw. inaktiviert werden.

Standardmäßig ist dieser Parameter auf ON gesetzt.

Sie können einzelne Funktionen der automatischen Verwaltung unabhängig voneinander aktivieren oder inaktivieren, indem Sie die folgenden Parameter definieren:

auto_db_backup
Dieser Parameter der automatischen Verwaltung aktiviert bzw. inaktiviert automatische Backup-Operationen für eine Datenbank. Eine Backup-Richtlinie (eine definierte Gruppe von Regeln oder Bestimmungen) kann zur Angabe der automatischen Funktionsweise verwendet werden. Die Backup-Richtlinie hat den Zweck, sicherzustellen, dass die Datenbank regelmäßig gesichert wird. Die Backup-Richtlinie für eine Datenbank wird automatisch erstellt, wenn der DB2-Diagnosemonitor zum ersten Mal ausgeführt wird. Standardmäßig ist dieser Parameter auf OFF gesetzt. Zur Aktivierung muss dieser Parameter auf ON gesetzt werden und der übergeordnete Parameter muss ebenfalls aktiviert sein.

auto_tbl_maint
Dieser Parameter ist das übergeordnete Element für alle Tabellenverwaltungsparameter (**auto_runstats**, **auto_stats_prof**, **auto_prof_upd** und **auto_reorg**). Wenn dieser Parameter inaktiviert wird, werden alle untergeordneten Parameter ebenfalls inaktiviert, jedoch ändern sich ihre Einstellungen, die in der Datenbankkonfigurationsdatei aufgezeichnet sind, nicht. Wenn dieser übergeordnete Parameter aktiviert wird, werden die aufgezeichneten Werte für die untergeordneten Parameter wirksam. Auf diese Weise kann die Tabellenverwaltung global aktiviert bzw. inaktiviert werden.

Standardmäßig ist dieser Parameter auf ON gesetzt.

auto_runstats
Dieser Parameter der automatischen Tabellenverwaltung aktiviert bzw. inaktiviert automatische **RUNSTATS**-Operationen für Tabellen in einer Datenbank. Eine **RUNSTATS**-Richtlinie (eine definierte Gruppe von Regeln oder Bestimmungen) kann zur Angabe der automatischen Funktionsweise verwendet werden. Die vom Dienstprogramm **RUNSTATS** erfassten Statistiken werden vom Optimierungsprogramm zur Bestimmung des effizientesten Plans für den Zugriff auf die physischen Daten verwendet. Zur Aktivierung muss dieser Parameter auf ON gesetzt werden und die übergeordneten Parameter müssen ebenfalls aktiviert sein.

Standardmäßig ist dieser Parameter auf ON gesetzt.

auto_stats_prof

Wichtig: Die automatische Statistikprofilerstellung gilt in Version 10.1 als veraltet und wird möglicherweise in einem zukünftigen Release entfernt. Weitere Informationen hierzu finden Sie im Abschnitt „Die automatische Statistikprofilerstellung gilt als veraltet" in *Neuerungen in DB2 Version 10.1*.

Dieser Parameter der automatischen Tabellenverwaltung aktiviert die Statistikprofilgenerierung, die dazu dient, Anwendungen zu unterstützen, deren Auslastungen komplexe Abfragen, zahlreiche Vergleichselemente, Joins und Gruppierungen unter Einbeziehung verschiedener Tabellen beinhalten. Zur Aktivierung muss dieser Parameter auf ON gesetzt werden und die übergeordneten Parameter müssen ebenfalls aktiviert sein.

Standardmäßig ist dieser Parameter auf OFF gesetzt.

Die automatische Statistikprofilgenerierung wird in Umgebungen mit partitionierten Datenbanken, in bestimmten Umgebungen mit föderierten Datenbanken, in DB2 pureScale-Umgebungen und bei aktivierter partitionsinterner Parallelität nicht unterstützt. Die automatische Generierung von Statistikprofilen kann nicht aktiviert werden, wenn der Datenbankkonfigurationsparameter **section_actuals** aktiviert ist (SQLCODE -5153).

auto_stmt_stats

Mit diesem Parameter wird die Erfassung von Echtzeitstatistiken aktiviert und inaktiviert. Dieser Parameter ist ein untergeordneter Parameter des Konfigurationsparameters **auto_runstats**. Diese Funktion wird nur aktiviert, wenn auch die übergeordnete Funktion, also der Konfigurationsparameter **auto_runstats**, aktiviert ist. Beispiel: Wenn **auto_stmt_stats** aktiviert werden soll, müssen **auto_maint**, **auto_tbl_maint** und **auto_runstats** auf ON gesetzt werden. Um den untergeordneten Wert beizubehalten, kann der Konfigurationsparameter **auto_runstats** auf ON gesetzt sein, während der Konfigurationsparameter **auto_maint** auf OFF gesetzt ist. Die entsprechende Funktion für die automatische Statistikerstellung ist weiterhin auf OFF gesetzt.

Unter der Annahme, dass die Funktionen für die automatische Statistikerstellung und für die automatische Reorganisation aktiviert sind, gelten die folgenden Einstellungen:

```
Automatische Verwaltung                          (AUTO_MAINT) = ON
  Automatisches Datenbankbackup                (AUTO_DB_BACKUP) = OFF
    Automatische Tabellenverwaltung            (AUTO_TBL_MAINT) = ON
      Automatische Statistikerstellung          (AUTO_RUNSTATS) = ON
        Echtzeitstatistikdaten                 (AUTO_STMT_STATS) = ON
        Statistiksichten                  (AUTO_STATS_VIEWS) = OFF
        Automatische Stichprobenentnahme (AUTO_SAMPLING) = OFF
Automatische Statistikprofilgener.   (AUTO_STATS_PROF) = OFF
        Statistikprofilaktualisierungen           (AUTO_PROF_UPD) = OFF
      Automatische Reorganisation                (AUTO_REORG) = ON
```

Sie können sowohl die Funktion für automatische Statistikerstellung als auch für die automatische Reorganisation vorübergehend inaktivieren; hierfür müssen Sie **auto_tbl_maint** auf OFF setzen. Beide Funktionen können später wieder aktiviert werden; hierfür müssen Sie **auto_tbl_maint** wieder auf ON setzen. Damit die Änderungen wirksam werden, ist es nicht notwendig, die Befehle **db2stop** oder **db2start** abzusetzen.

Standardmäßig ist dieser Parameter auf ON gesetzt.

auto_stats_views

Dieser Parameter aktiviert und inaktiviert die automatische Statistikerfassung für Statistiksichten. Die Statistiken für Statistiksichten werden automatisch verwaltet, wenn dieser Parameter auf ON gesetzt ist.

Standardmäßig ist dieser Parameter auf OFF gesetzt.

auto_prof_upd

Wichtig: Die automatische Statistikprofilerstellung gilt in Version 10.1 als veraltet und wird möglicherweise in einem zukünftigen Release entfernt. Weitere Informationen hierzu finden Sie im Abschnitt „Die automatische Statistikprofilerstellung gilt als veraltet" in *Neuerungen in DB2 Version 10.1*.

Wenn dieser Parameter der automatischen Tabellenverwaltung (ein untergeordneter Parameter von **auto_stats_prof**) aktiviert ist, gibt er an, dass das **RUNSTATS**-Profil mit Empfehlungen zu aktualisieren ist. Wenn dieser Parameter inaktiviert ist, werden Empfehlungen in der Tabelle opt_feedback_ranking gespeichert, die Sie einsehen können, wenn Sie das **RUNSTATS**-Profil manuell aktualisieren. Zur Aktivierung muss dieser Parameter auf ON gesetzt werden und die übergeordneten Parameter müssen ebenfalls aktiviert sein.

Standardmäßig ist dieser Parameter auf OFF gesetzt.

auto_reorg

Dieser Parameter der automatischen Tabellenverwaltung aktiviert bzw. inaktiviert die automatische Tabellen- und Indexreorganisation für eine Datenbank. Eine Reorganisationsrichtlinie (eine definierte Gruppe von Regeln oder Bestimmungen) kann zur Angabe der automatischen Funktionsweise verwendet werden. Zur Aktivierung muss dieser Parameter auf ON gesetzt werden und die übergeordneten Parameter müssen ebenfalls aktiviert sein.

Standardmäßig ist dieser Parameter auf OFF gesetzt.

auto_sampling

Dieser Parameter steuert, ob bei der automatischen Statistikerfassung beim Erfassen von Statistikdaten für eine große Tabelle die Stichprobenentnahme verwendet werden kann. Zum Aktivieren der automatischen Stichprobenentname legen Sie für **auto_maint**, **auto_tbl_maint**, **auto_runstats** und **auto_sampling** die Einstellung ON fest. Wenn die automatische Statistikerfassung aktiviert ist, wird die Stichprobenrate automatisch festgelegt.

Standardmäßig ist dieser Parameter auf OFF gesetzt.

auto_reval - Automatische Reaktivierung und Inaktivierung (Konfigurationsparameter)

Dieser Konfigurationsparameter steuert die Semantik für die Reaktivierung und Inaktivierung.

Konfigurationstyp

Datenbank

Parametertyp

- Online konfigurierbar
- Nach Member in einer DB2 pureScale-Umgebung konfigurierbar

Standardwert [Bereich]

DEFERRED [IMMEDIATE, DISABLED, DEFERRED, DEFERRED_FORCE]

Wenn Sie eine neue Datenbank erstellen, wird dieser Parameter auf den Standardwert DEFERRED gesetzt.

Wenn Sie ein Upgrade einer Datenbank von Version 9.5 oder einer früheren Version durchführen, wird der Parameter **auto_reval** auf den Wert DISABLED gesetzt. Das Verhalten für die Reaktivierung (erneute Gültigmachung) hat sich gegenüber früheren Releases nicht geändert.

Wenn Sie diesen Parameter auf den Wert IMMEDIATE setzen, bedeutet dies, dass alle abhängigen Objekte unverzüglich nach ihrer Inaktivierung wieder reaktiviert werden. Diese gilt für einige DDL-Anweisungen wie ALTER TABLE, ALTER COLUMN, oder CREATE OR REPLACE. Die erfolgreiche Reaktivierung der abhängigen Objekte ist von keinen anderen DDL-Änderungen abhängig und kann daher unverzüglich ausgeführt werden.

Wenn Sie diesen Parameter auf den Wert DEFERRED setzen, bedeutet dies, dass alle abhängigen Objekte reaktiviert werden, wenn das nächste Mal auf diese Objekte zugegriffen wird.

Beachten Sie, dass, wenn dieser Parameter den Wert IMMEDIATE oder DEFERRED hat und eine Reaktivierungsoperation fehlschlägt, ungültige abhängige Objekte ungültig bleiben, bis das nächste Mal auf sie zugegriffen wird.

Wenn Sie diesen Parameter auf den Wert DEFERRED_FORCE setzen, entspricht das Verhalten dem des Werts DEFERRED, jedoch wird eine zusätzliche Funktion für CREATE-Operationen mit Fehlern aktiviert.

In einigen Fällen kann die Syntax, die Sie explizit angeben, die Einstellung des Parameters **auto_reval** außer Kraft setzen. Wenn Sie zum Beispiel die Klausel DROP COLUMN der Anweisung ALTER TABLE ohne Angabe von CASCADE oder RESTRICT verwenden, wird die Semantik durch den Parameter **auto_reval** gesteuert. Wenn Sie jedoch CASCADE oder RESTRICT angeben, wird die vorherige CASCADE- bzw. RESTRICT-Semantik verwendet. Dies überschreibt die neue Semantik, die durch den Parameter **auto_reval** angegeben wird.

Dieser Konfigurationsparameter ist dynamisch, das heißt, eine Änderung seines Werts wird unverzüglich wirksam. Sie brauchen keine erneute Verbindung zur Datenbank herzustellen, um die Änderung in Kraft zu setzen.

autorestart - Automatischer Neustart aktiviert

Der Parameter **autorestart** legt fest, ob der Datenbankmanager die Recovery nach Absturz automatisch einleitet, wenn ein Benutzer die Verbindung zu einer Datenbank herstellt, die zuvor abnormal beendet wurde. Wenn der Parameter **autorestart** nicht definiert ist, muss der Benutzer explizit den Befehl RESTART DATABASE absetzen, bevor eine Verbindung zu der Datenbank hergestellt werden kann.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
On [On; Off]

Wenn der Parameter **autorestart** auf ON gesetzt ist, wird beim nächsten Verbindungsversuch mit der Datenbank die automatische Recovery nach Absturz ausgeführt (falls erforderlich).

In einer DB2 pureScale-Umgebung ändert sich das Verhalten beim automatischen Neustart. Wenn ein Datenbankmember fehlschlägt, wird die Recovery nach Absturz für dieses Member automatisch eingeleitet. Wenn beide Cluster-Caching-Funktionen gleichzeitig fehlschlagen, wird automatisch die Recovery nach Absturz einer Gruppe eingeleitet. Während der Recovery nach Absturz eines Members werden keine Verbindungen zur Datenbank über dieses Member zugelassen. Ein Fehler SQL1015N wird zurückgegeben, wenn versucht wird, eine Verbindung zur Datenbank über ein Member herzustellen, für das eine Recovery nach Absturz erforderlich ist. Wenn die Recovery nach Absturz fehlschlägt, wird erneut versucht, eine Recovery nach Absturz durchzuführen. Schlägt auch der zweite Versuch für die Recovery nach Absturz fehl, wird ein ALERT für das Member auf dem Host gesetzt, auf dem die Recovery fehlgeschlagen ist, und ein Light-Neustart für dieses Member wird auf einem anderen Host ausgeführt.

Wenn der Parameter **autorestart** auf OFF gesetzt ist, sind die Prozesse für automatische Recovery nach Absturz eines Members und für Recovery nach Absturz einer Gruppe inaktiviert, d. h. sie müssen bei Bedarf manuell ausgeführt werden. Die Prozesse für Recovery nach Absturz können mit dem Befehl **RESTART DATABASE** eingeleitet werden.

Wenn ein Member in einer DB2 pureScale-Umgebung fehlschlägt und auf seinem ursprünglichen Host (auch als Benutzerhost bezeichnet) nicht erneut gestartet werden kann, führt DB2-Cluster-Services auf einem der anderen verfügbaren Member-Hosts in dem DB2 pureScale-Cluster einen Neustart dieses Members aus. Diese Vorgehensweise wird als Light-Neustart bezeichnet. Wenn sich ein Member im Modus für Light-Neustart befindet, können Sie keine direkte Verbindung zu diesem Member herstellen, d. h. es kann kein manueller Neustart ausgeführt werden. Sie müssen den Parameter **autorestart** vorher wieder auf ON setzen. Danach erkennt das Member im Modus für Light-Neustart automatisch den geänderten Datenbankkonfigurationsparameter und leitet bei Bedarf die Recovery nach Absturz ein.

avg_appls - Durchschnittliche Anzahl aktiver Anwendungen

Mithilfe dieses Parameters versucht das Abfrageoptimierungsprogramm zu ermitteln, wie viel Platz im Pufferpool zur Laufzeit für den ausgewählten Zugriffsplan verfügbar ist.

Konfigurationstyp
Datenbank

Parametertyp

- Online konfigurierbar
- Nach Member in einer DB2 pureScale-Umgebung konfigurierbar

Weitergabeklasse
Anweisungsgrenzwert

Standardwert [Bereich]
Automatic [1 – maxappls]

Maßeinheit
Zähler

Empfehlung: Wenn das DB2-Datenbankprodukt in einer Mehrbenutzerumgebung ausgeführt wird, kann es vorteilhaft sein, wenn das Abfrageoptimierungsprogramm weiß, dass mehrere Abfragebenutzer das System verwenden (besonders bei komplexen Abfragen und einem großen Pufferpool), sodass das Optimierungsprogramm in seinen Annahmen zur Verfügbarkeit von Pufferpool weniger großzügig ist.

Wenn dieser Parameter auf den Wert AUTOMATIC gesetzt wird, wird er sofort mit einem geeigneten Wert aktualisiert, wenn die Datenbankkonfigurationsdatei erstellt oder zurückgesetzt wird.

Das Festlegen dieses Parameters kann dazu beitragen, dass das Optimierungsprogramm die Belegung des Pufferpools genauer modellieren kann. Wenn Sie diesen Parameter manuell festlegen, müssen Sie mit dem Wert 2 beginnen (unabhängig von der durchschnittlichen Zahl ausgeführter Anwendungen). Nach dem Bewerten des Verhaltens des Optimierungsprogramms und dem Testen der Leistung der Anwendung bei dieser Einstellung können Sie den Wert des Parameters in kleinen Schritten erhöhen. Setzen Sie die Bewertung des Verhaltens des Optimierungsprogramms und das Testen der Leistung der Anwendung jedes Mal fort, wenn Sie den Wert des Parameters erhöhen. Wenn dieser Parameter auf einen zu hohen Wert gesetzt wird, kann dies dazu führen, dass das Optimierungsprogramm die Menge des Platzes im Pufferpool unterschätzt, die für die Abfrage verfügbar ist.

Nach dem Ändern des Werts sollten Sie das erneuten Binden der Anwendungen mit dem Befehl **REBIND PACKAGE** in Betracht ziehen.

backup_pending - Backup anstehend

Der Parameter **backup_pending** weist darauf hin, dass Sie ein Gesamtbackup der Datenbank ausführen müssen, bevor Sie auf sie zugreifen.

Konfigurationstyp
Datenbank

Parametertyp
Informativ

Dieser Parameter wird nur auf ON gesetzt, wenn die Datenbankkonfiguration geändert wird und die Datenbank anschließend wiederherstellbar ist. Das bedeutet, die Parameter **logarchmeth1** und **logarchmeth2** waren zunächst auf OFF gesetzt, später wird jedoch einer der Parameter (oder beide) gesetzt und die Aktualisierung der Datenbankkonfiguration wird akzeptiert.

blk_log_dsk_ful - Bei voller Protokollplatte blockieren

Dieser Parameter kann definiert werden, um zu verhindern, dass Fehler aufgrund erschöpfter Festplattenkapazität generiert werden, wenn das DB2-Datenbanksystem keine neue Protokolldatei im aktiven Protokollpfad erstellen kann.

Konfigurationstyp
Datenbank

Parametertyp

- Online konfigurierbar
- Nach Member in einer DB2 pureScale-Umgebung konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
No [Yes; No]

Anstatt einen Fehler aufgrund erschöpfter Festplattenkapazität zu generieren, versucht das DB2-Datenbanksystem alle fünf Minuten, die Protokolldatei zu erstellen, bis diese Datei erfolgreich erstellt wurde. Nach jedem Versuch schreibt das DB2-Datenbanksystem eine Nachricht in das Protokoll mit den Benachrichtigungen für die Systemverwaltung. Das Überwachen dieses Protokolls mit Benachrichtigung für die Systemverwaltung stellt die einzige Möglichkeit dar, zu bestätigen, dass eine Anwendung blockiert ist. Bis zur erfolgreichen Erstellung der Protokolldatei kann keine Benutzeranwendung, die eine Aktualisierung von Tabellendaten ausführen will, eine Transaktion festschreiben. Auf Lesezugriff beschränkte Abfragen sind u. U. nicht direkt betroffen. Wenn jedoch eine Abfrage auf Daten zugreifen muss, die aufgrund einer Aktualisierungsanforderung gesperrt sind, oder auf eine Datenseite, die im Pufferpool von der aktualisierenden Anwendung korrigiert wird, erscheinen auch auf Lesezugriff beschränkte Abfragen blockiert.

Wenn **blk_log_dsk_ful** auf yes gesetzt wird, werden die Anwendungen blockiert, sobald das DB2-Datenbanksystem einen Fehler aufgrund erschöpfter Festplattenkapazität feststellt. Dadurch können Sie den Fehler beheben und die Transaktion abschließen. Die Fehlersituation aufgrund erschöpfter Festplattenkapazität können Sie beheben, indem Sie entweder die Kapazität des Dateisystems vergrößern oder externe Dateien aus dem Dateisystem entfernen.

Wenn **blk_log_dsk_ful** auf no gesetzt ist und eine Transaktion einen Fehler aufgrund erschöpfter Festplattenkapazität feststellt, schlägt die betreffende Transaktion fehl und wird mit ROLLBACK rückgängig gemacht.

Setzen Sie **blk_log_dsk_ful** auf yes, wenn die unbegrenzte Protokollierung verwendet wird (d. h. wenn der Datenbankkonfigurationsparameter **logsecond** auf -1 gesetzt ist). Bei der unbegrenzten Protokollierung kann die Datenbank in manchen Situationen offline gehen, wenn eine Transaktion einen Fehler aufgrund erschöpfter Festplattenkapazität verursacht.

blocknonlogged - Erstellung von Tabellen blockieren, die nicht protokollierte Aktivitäten zulassen

Dieser Parameter gibt an, ob es der Datenbankmanager zulässt, dass für Tabellen die Attribute NOT LOGGED oder NOT LOGGED INITIALLY aktiviert werden.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Nach Member in einer DB2 pureScale-Umgebung konfigurierbar

Standardwert [Bereich]
No [Yes, No]

Standardmäßig ist der Parameter **blocknonlogged** auf den Wert NO gesetzt: Nicht protokollierte Operationen werden zugelassen und Sie verfügen über die Leistungsvorteile, die durch die verringerte Protokollierung erzielt werden. Bei dieser Konfiguration sind jedoch einige potenzielle Nachteile, insbesondere in HADR-Datenbankumgebungen (High Availability Disaster Recovery), zu beachten. DB2-HADR-Datenbankumgebungen arbeiten mit Datenbankprotokollen, um Daten aus der primären Datenbank in die Bereitschaftsdatenbank zu replizieren. Nicht proto-

kollierte Operationen sind in der Primärdatenbank zwar zulässig, werden aber nicht in die Bereitschaftsdatenbank repliziert. Wenn Sie nicht protokollierte Operationen in der Primärdatenbank ausführen, muss die Bereitschaftsdatenbank reinitialisiert werden. Sie können zum Beispiel Onlineteilungen von Spiegeldatenbanken oder die Unterstützung für ausgesetzte Ein-/Ausgabe verwenden, um die Bereitschaftsdatenbank nach nicht protokollierten Operationen zu resynchronisieren.

Hinweise

- Wenn für **blocknonlogged** die Einstellung YES definiert wird, schlagen die Anweisungen CREATE TABLE und ALTER TABLE fehl, falls eine der folgenden Bedingungen vorliegt:
  - Der Parameter NOT LOGGED INITIALLY wird angegeben.
  - Der Parameter NOT LOGGED wird für eine LOB-Spalte angegeben.
  - Eine CLOB-, DBCLOB- oder BLOB-Spalte wird als nicht protokolliert definiert.
- Wenn für **blocknonlogged** die Einstellung YES definiert wird, schlägt der Befehl LOAD fehl, wenn die folgenden Situationen vorliegen:
  - Die Option NONRECOVERABLE wird angegeben.
  - Die Option COPY NO wird angegeben.

cf_catchup_trgt - Catch-up-Sollzeit für sekundäre Cluster-Caching-Funktion (Konfigurationsparameter)

Dieser Konfigurationsparameter gibt die Sollzeit in Minuten bis zum Abschluss der Ausgleichung (Catch-up) an, durch die eine neu hinzugefügte bzw. erneut gestartete Cluster-Caching-Funktion in den PEER-Status mit einer vorhandenen primären Cluster-Caching-Funktion gebracht wird.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
15 [1 – 600]

Maßeinheit
Minuten seit dem Starten der sekundären Cluster-Caching-Funktion

Dieser Parameter gibt die Sollzeit an, in der Member eine neu hinzugefügte sekundäre Cluster-Caching-Funktion in den PEER-Status mit der primären Cluster-Caching-Funktion bringen soll. Sobald der PEER-Status hergestellt ist, kann die sekundäre Cluster-Caching-Funktion die Rolle der primären Cluster-Caching-Funktion übernehmen, wenn die primäre Cluster-Caching-Funktion ausfällt oder für Wartungszwecke abgeschaltet wird. Ein niedriger Wert für **cf_catchup_trgt** führt dazu, dass Member häufig Catch-up-Operationen ausführen und minimiert das Zeitfenster, in dem eine DB2 pureScale-Instanz ohne eine sekundäre Cluster-Caching-Funktion auskommen muss, die die Primärrolle ersatzweise übernehmen kann. Ein niedriger Wert wirkt sich stärker auf Datenbanktransaktionen und auf die Gesamtleistung des Systems aus, da mehr Systemressourcen (z. B. E/A-Bandbreite) für die Catch-up-Aktivität verwendet werden.

Anmerkung: Obwohl die Standardeinstellung für **cf_catchup_trgt** einen Ausgleich zwischen Systemleistung und hoher Verfügbarkeit schafft, kann die folgende Faustregel zum Optimieren dieses Parameters verwendet werden:

- Wenn die Verfügbarkeit Vorrang vor der Leistung hat, verwenden Sie einen niedrigen Wert für **cf_catchup_trgt**.
- Wenn die Leistung Vorrang vor der Verfügbarkeit hat, verwenden Sie einen hohen Wert für **cf_catchup_trgt**.

cf_db_mem_sz - Datenbankspeicher (Konfigurationsparameter)

Dieser Parameter steuert den Gesamtspeichergrenzwert für die Cluster-Caching-Funktion (auch als CF bezeichnet) für diese Datenbank.

Konfigurationstyp
Datenbank

Gilt für
Gilt nur für eine DB2 pureScale-Instanz.
- Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
AUTOMATIC [32768 - 4 294 967 295]

Maßeinheit
Seiten (4 KB)

Die Summe der Cluster-Caching-Funktion-Strukturspeicherbegrenzungen für die Parameter **cf_gbp_sz**, **cf_lock_sz** und **cf_sca_sz** muss kleiner als die CF-Strukturspeicherbegrenzung für den Parameter **cf_db_mem_sz** sein. Die automatische Speicheroptimierung zwischen CF-Ressourcen wie Gruppenpufferpool (Group Buffer Pool, GBP), gemeinsamer Kommunikationsbereich (Shared Communication Area, SCA) und Sperrstrukturen innerhalb derselben Datenbank ist an den Parameter **cf_db_mem_sz** gebunden.

Eine automatische Leistungsoptimierung von CF-Speicher zwischen Datenbanken wird nicht unterstützt.

Hinweise zur Verwendung

- Geben Sie für den Konfigurationsparameter **numdb** einen Wert an, der größer oder gleich der Gesamtzahl der Datenbanken in der Instanz ist. Geben Sie außerdem für **DB2_DATABASE_CF_MEMORY** einen Wert an, der zulässt, dass alle Datenbanken in dieser Instanz gleichzeitig aktiv sein können (unabhängig davon, ob sie normalerweise inaktiv oder aktiv sind).
- Wenn der Wert für **cf_db_mem_sz** manuell angegeben wird, muss er kleiner sein als die Gesamtgröße des CF-Serverspeichers. Die Gesamtgröße des CF-Serverspeichers wird durch den Konfigurationsparameter **cf_mem_sz** des Datenbankmanagers gesteuert.
- Wenn der vorherige Wert des Parameters **cf_db_mem_sz** auf AUTOMATIC gesetzt war, kann der aktuelle Parameterwert mit der Option **SHOW DETAIL** des Befehls **GET DATABASE CONFIGURATION** ermittelt werden.
- Die Verwendung des Konfigurationsparameters **cf_db_mem_sz** muss mit der Registrierdatenbankvariablen **DB2_DATABASE_CF_MEMORY** und dem Konfigurationsparameter **numdb** abgeglichen werden.

Einschränkungen

- 3840 zusätzlich erforderliche 4-KB-Seiten werden vom Parameterwert **cf_mem_sz** zur internen Verwendung durch die CF abgezogen. Für jede aktive Datenbank in der Instanz werden zusätzlich 256 4-KB-Seiten vom Parameterwert **cf_mem_sz** reserviert. Diese zusätzlichen Seiten müssen nur berücksichtigt werden, wenn der Parameter **cf_mem_sz** manuell festgelegt wird.
- Beim Ändern des festen Werts für diesen Parameter in einen anderen Wert, der *kleiner* ist als der aktuelle Wert, muss der neue Wert größer als die Summe der Speichergrößen für die Strukturparameter **cf_gbp_sz**, **cf_lock_sz** und **cf_sca_sz** sein. Andernfalls schlägt die Operation fehl. Um dieses Problem zu vermeiden, reduzieren Sie die Größe der einzelnen Strukturspeicher, bevor Sie versuchen, diesen Parameterwert zu verkleinern.
- Beim Ändern des festen Werts für diesen Parameter in einen neuen Wert, der *größer* ist als der aktuelle Wert, gilt eine Obergrenze, zusätzlich zur Summe der Größen des CF-Serverspeichers, die im Parameter **cf_mem_sz** definiert sind. In der Regel wird diese Obergrenze durch einen Speicherpuffer mit 3600 bis 5000 Seiten definiert, der zwischen den Parameterwerten für **cf_db_mem_sz** und für **cf_mem_sz** erforderlich ist. Der Parameterwert für **cf_db_mem_sz** sollte nicht größer sein als die Differenz zwischen **cf_mem_sz** und diesem Speicherpuffer.

cf_gbp_sz - Gruppenpufferpool (Konfigurationsparameter)

Dieser Parameter legt die Speicherkapazität fest, die von der Cluster-Caching-Funktion (auch als CF bezeichnet) für Gruppenpufferpools (GBP) in dieser Datenbank verwendet wird.

Konfigurationstyp
Datenbank

Gilt für
Gilt nur für eine DB2 pureScale-Instanz.
- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
AUTOMATIC [AUTOMATIC, 4096 - 4294967296]

Maßeinheit
Seiten (4 KB)

Zuordnung
Bei der ersten Verbindungsherstellung oder Aktivierung der Datenbank auf einem DB2-Member

Freigabe
Beim Inaktivieren der Datenbank auf dem letzten DB2-Member

Der GBP enthält zwei Haupttypen von Ressourcen: Verzeichniseinträge und Datenelemente. Ein Verzeichniseintrag speichert zugehörige Metadateninformationen für eine Seite. Ein Datenelement speichert die Seitendaten an sich.

DB2-Member führen die Zwischenspeicherung von Seiten weiterhin in ihren eigenen lokalen Pufferpools aus. Der Gruppenpufferpool wird von den lokalen Puffer-

managern nur verwendet, um die systemübergreifende Pufferkohärenz zu erhalten, und kann nicht direkt von DB2-EDUs referenziert werden, die auf einem beliebigen DB2-Member ausgeführt werden.

cf_lock_sz - CF-Sperrenmanager (Konfigurationsparameter)

Dieser Parameter legt die Speicherkapazität fest, die von der CF für Sperren in dieser Datenbank verwendet wird.

Konfigurationstyp
Datenbank

Gilt für
Gilt nur für eine DB2 pureScale-Instanz.

- Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
AUTOMATIC [AUTOMATIC, 4096 - 1073741824]

Maßeinheit
Seiten (4 KB)

Zuordnung
Bei der ersten Aktivierung der Datenbank auf einem Member.

Anmerkung: Speicher wird in der CF zugeordnet und nicht in einem der Member. Wenn mehr als eine CF vorhanden ist, ordnet jede CF die gleiche Speicherkapazität zu.

Freigabe
Wenn die Datenbank in allen Membern inaktiviert wird.

Hierbei handelt es sich um einen globalen Konfigurationsparameter, d. h. für jedes Member der DB2 pureScale-Instanz wird derselbe Wert verwendet.

Hinweis: Der Speicher in der CF wird nur einmal zugeordnet (beim ersten Aktivieren der Datenbank in einem beliebigen Member). Die Belegung des CF-Sperrenspeichers kann mit dem Monitorelement **current_cf_lock_size** überwacht werden.

CF_sca_sz - Gemeinsamer Kommunikationsbereich (Konfigurationsparameter)

Dieser Konfigurationsparameter für den gemeinsamen Kommunikationsbereich (Shared Communication Area, SCA) legt fest, welche Speicherkapazität von SCA in der Cluster-Caching-Funktion (CF) verwendet wird. Der gemeinsame Kommunikationsbereich (SCA) ist eine Einheit auf Datenbankebene, die datenbankweite Steuerblockdaten für Tabellen, Indizes, Tabellenbereiche und Kataloge enthält.

Konfigurationstyp
Datenbank

Gilt für
DB2 pureScale-Umgebung

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
AUTOMATIC [AUTOMATIC, 2048 – 1073741824]

Maßeinheit
Seiten (4 KB)

Zuordnung
Bei der ersten Aktivierung der Datenbank auf einem DB2-Member

Freigabe
Wenn die Datenbank gelöscht oder der CF-Server gestoppt wird

Wenn Sie den Parameter **cf_sca_sz** auf den Standardwert AUTOMATIC setzen, wird beim ersten Aktivieren der Datenbank auf einem beliebigen Member vom DB2-Datenbankmanager ein Speicherwert berechnet, der für grundlegende Datenbankoperationen ausreicht. Wenn Sie die genaue Speichergröße für den SCA konfigurieren möchten, können Sie den Parameter **cf_sca_sz** mit einem festen numerischen Wert definieren.

Den aktuellen Wert für die SCA-Größe können Sie durch Ausführen des Befehls **GET DB CFG SHOW DETAIL** abrufen. Wenn der Parameter **cf_sca_sz** auf AUTOMATIC gesetzt ist, zeigt die Klausel SHOW DETAIL den berechneten Wert sowie den zugeordneten Wert für den SCA an.

Wenn der SCA-Speicher durch Datenbankoperationen vollständig ausgelastet ist, wird eine Fehlernachricht für Speicherengpass zurückgegeben. In diesem Fall können Sie den SCA-Speicher vergrößern, indem Sie einen höheren Wert für den Parameter **cf_sca_sz** angeben.

Weitere Details finden Sie in „Speicher der Cluster-Caching-Funktion für eine Datenbank konfigurieren".

catalogcache_sz - Katalogcachegröße

Mit diesem Parameter wird der maximale Speicher in Seiten angegeben, die der Katalogcache vom Datenbankzwischenspeicher verwenden kann.

Konfigurationstyp
Datenbank

Parametertyp

- Online konfigurierbar
- Nach Member in einer DB2 pureScale-Umgebung konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
-1 [**maxappls***5, 8 - 524 288]

Anmerkung: Der Standardwert unterliegt nach der ursprünglichen Datenbankerstellung Änderungen durch den DB2 Configuration Advisor.

Maßeinheit
Seiten (4 KB)

Zuordnung
Wenn die Datenbank initialisiert wird

Freigabe
Wenn die Datenbank heruntergefahren wird

Dieser Parameter wird aus dem gemeinsam genutzten Datenbankspeicher zugeordnet und für das Caching (Zwischenspeichern) von Systemkataloginformationen verwendet. In einem partitionierten Datenbanksystem gibt es für jede Datenbankpartition einen Katalogcache.

Das Caching von Kataloginformationen in einzelnen Datenbankpartitionen ermöglicht es dem Datenbankmanager, die Verarbeitungszeit zu verringern, da auf die Systemkataloge (oder auf den Katalogknoten in einer Umgebung mit partitionierten Datenbanken) zum Abrufen von Informationen, die bereits abgerufen wurden, nicht mehr zugegriffen werden muss. Das Verwenden des Katalogcache kann helfen, die Gesamtleistung folgender Operationen und Anwendungen zu erhöhen:

- Binden von Paketen und Kompilieren von SQL- und XQuery-Anweisungen
- Operationen, bei denen Zugriffsrechte auf Datenbankebene, Routinenzugriffsrechte, Zugriffsrechte für globale Variablen und Rollenberechtigungen überprüft werden
- Anwendungen, die mit Nicht-Katalogknoten in einer Umgebung mit partitionierten Datenbanken verbunden sind

Wenn der Standardwert (-1) für diesen Parameter in einer Serverumgebung oder in einer Umgebung mit partitionierten Datenbanken gesetzt ist, wird als Wert für die Berechnung der Seitenzuordnung das Fünffache des für den Konfigurationsparameter **maxappls** angegebenen Werts verwendet. Wenn das Fünffache von **maxappls** jedoch kleiner als 8 ist, gilt dies nicht. In diesem Fall setzt der Standardwert -1 den Parameter **catalogcache_sz** auf den Wert 8.

Empfehlung: Verwenden Sie zu Beginn den Standardwert und optimieren Sie den Wert mithilfe des Datenbanksystemmonitors. Beim Optimieren dieses Parameters sollten Sie überlegen, ob der zusätzliche Speicher, der für den Katalogcache reserviert wird, möglicherweise besser für einen anderen Zweck zugeordnet werden sollte, z. B. für den Pufferpool oder den Paketcache.

Eine Optimierung dieses Parameters ist besonders wichtig, wenn die Auslastung für kurze Zeit viele SQL- oder XQuery-Kompilierungen umfasst und anschließend nur noch wenige oder gar keine Kompilierungen stattfinden. Wenn der Cache zu groß ist, kann Speicher für Kopien nicht mehr benötigter Informationen verschwendet werden.

Überlegen Sie, ob in einer Umgebung mit partitionierten Datenbanken für **catalogcache_sz** auf dem Katalogknoten ein größerer Wert festgelegt werden muss, da die auf Nicht-Katalogknoten benötigten Kataloginformationen immer zuerst auf dem Katalogknoten zwischengespeichert werden.

Mithilfe der Monitorelemente **cat_cache_lookups** (Suchfunktionen des Katalogcache), **cat_cache_inserts** (Einfügungen im Katalogcache), **cat_cache_overflows** (Überläufe des Katalogcache) und **cat_cache_size_top** (Obere Grenze des Katalogcache) können Sie ermitteln, ob dieser Konfigurationsparameter angepasst werden sollte.

Anmerkung: Der Katalogcache ist auf allen Knoten in einer Umgebung mit partitionierten Datenbanken vorhanden. Da für jeden Knoten eine lokale Datenbankkonfigurationsdatei vorhanden ist, definiert der Wert des Parameters **catalogcache_sz** für jeden Knoten die Größe des lokalen Katalogcache. Um ein effizientes Caching zu gewährleisten und Überlaufszenarios zu vermeiden, müssen Sie den Wert für **catalogcache_sz** auf jedem Knoten explizit festlegen und dabei die Möglichkeit in Betracht ziehen, den Wert für **catalogcache_sz** auf Nicht-Katalogknoten kleiner zu

wählen als auf Katalogknoten. Bedenken Sie, dass die Informationen, die auf Nicht-Katalogknoten zwischengespeichert werden müssen, aus dem Cache des Katalogknotens abgerufen werden. Ein Katalogcache auf Nicht-Katalogknoten ist wie eine Untermenge der Informationen des Katalogcache auf dem Katalogknoten.

Im Allgemeinen ist ein größerer Cachespeicher erforderlich, wenn eine UOW (Unit of Work, Arbeitseinheit) dynamische SQL- oder XQuery-Anweisungen enthält oder wenn Sie Pakete binden, die eine große Anzahl statischer SQL- oder XQuery-Anweisungen enthalten.

chngpgs_thresh - Schwellenwert für geänderte Seiten

Dieser Parameter gibt den Prozentsatz der geänderten Seiten an, bei dem die asynchronen Seitenlöschfunktionen gestartet werden, wenn sie zum gegebenen Zeitpunkt nicht aktiv sind.

Konfigurationstyp
Datenbank

Parametertyp

- Konfigurierbar
- Nach Member in einer DB2 pureScale-Umgebung konfigurierbar

Standardwert [Bereich]
`60 [5 – 99 ]`

Anmerkung: Der Standardwert unterliegt nach der ursprünglichen Datenbankerstellung Änderungen durch den DB2 Configuration Advisor.

Maßeinheit
Prozent

Asynchrone Seitenlöschfunktionen schreiben geänderte Seiten aus dem Pufferpool (oder den Pufferpools) auf Platte, bevor der Bereich im Pufferpool von einem Datenbankagenten angefordert wird. Daher müssen Datenbankagenten in der Regel nicht die Auslagerung geänderter Seiten abwarten, bevor sie den Speicherbereich im Pufferpool nutzen können. Dadurch wird die Gesamtleistung der Datenbankanwendungen verbessert.

Wenn die Seitenlöschfunktionen gestartet werden, erstellen sie eine Liste der Seiten, die auf Platte zu schreiben sind. Wenn das Schreiben dieser Seiten auf Platte abgeschlossen ist, werden die Seitenlöschfunktionen wieder inaktiv und warten auf den nächsten Starttrigger.

Wenn die Registrierdatenbankvariable **DB2_USE_ALTERNATE_PAGE_CLEANING** definiert ist (d. h., die alternative Methode der Seitenbereinigung verwendet wird), hat der Parameter **chngpgs_thresh** keine Wirkung und der Datenbankmanager bestimmt automatisch, wie viele genutzte Seiten im Pufferpool zu behalten sind.

Empfehlung: Bei Datenbanken mit hohem Aufkommen an aktualisierenden Transaktionen können Sie allgemein sicherstellen, dass genügend freie Seiten im Pufferpool verfügbar sind, indem Sie diesen Parameter auf einen Wert setzen, der gleich groß wie oder kleiner als der Standardwert ist. Ein Prozentsatz über dem Standardwert kann sich positiv auf die Leistung auswirken, wenn in Ihrer Datenbank eine kleine Anzahl sehr großer Tabellen gespeichert ist.

codepage - Codepage für die Datenbank

Dieser Parameter zeigt die Codepage an, die zur Erstellung der Datenbank verwendet wurde. Der Wert des Parameters **codepage** wird auf der Basis des Werts für den Parameter **codeset** abgeleitet.

Konfigurationstyp
Datenbank

Parametertyp
Informativ

codeset - Codierter Zeichensatz für die Datenbank

Dieser Parameter zeigt den codierten Zeichensatz an, der zur Erstellung der Datenbank verwendet wurde. Der codierte Zeichensatz wird vom Datenbankmanager zur Bestimmung des Parameters **codepage** verwendet.

Konfigurationstyp
Datenbank

Parametertyp
Informativ

collate_info - Informationen zur Sortierfolge

Dieser Parameter legt die Sortierfolge der Datenbank fest. Bei einer sprachsensitiven Sortierfolge enthalten die ersten 256 Byte die Zeichenfolgedarstellung des Namens der Sortierfolge (z. B. "SYSTEM_819_US").

Dieser Parameter kann nur mit der API db2CfgGet angezeigt werden. Er kann **nicht** mithilfe des Befehlszeilenprozessors angezeigt werden.

Konfigurationstyp
Datenbank

Parametertyp
Informativ

Dieser Parameter enthält 260 Byte mit Informationen zur Sortierfolge der Datenbank. Die ersten 256 Byte geben die Sortierfolge der Datenbank an, wobei Byte „n" die Sortierwertigkeit des Codepunkts enthält, dessen zugrundeliegende dezimale Darstellung „n" in der Codepage der Datenbank ist.

Die letzten 4 Byte enthalten interne Informationen zur Art der Sortierfolge. Bei den letzten vier Byte des Parameters handelt es sich um eine ganze Zahl. Die ganze Zahl ist abhängig von der Endian-Folge der Plattform. Folgende Werte sind möglich:

- **0** – Die Sortierfolge enthält nicht eindeutige Wertigkeiten.
- **1** – Die Sortierfolge enthält ausschließlich eindeutige Wertigkeiten.
- **2** – Die Sortierfolge ist die Identitätssortierfolge, nach der Zeichenfolgen Byte für Byte verglichen werden.
- **3** – Die Sortierfolge lautet NLSCHAR und wird für das Sortieren von Zeichen in der Datenbank für Thailändisch TIS620-1 (Codepage 874) verwendet.
- **4** – Die Sortierfolge lautet IDENTITY_16BIT und implementiert den Algorithmus „CESU-8 Compatibility Encoding Scheme for UTF–16: 8–Bit", wie er im Unicode Technical Report 26 angegeben wird, der auf der Website des Unicode Technical Consortium unter der folgenden Adresse zur Verfügung steht: http://www.unicode.org.

- **X'8001'** – Die Sortierfolge lautet UCA400_NO und implementiert den UCA-Algorithmus (Unicode Collation Algorithm) auf der Grundlage von Unicode Standard Version 4.0.0 mit implizit aktivierter Normalisierung.
- **X'8002'** – Die Sortierfolge lautet UCA400_LTH und implementiert den UCA-Algorithmus (Unicode Collation Algorithm) auf der Grundlage von Unicode Standard Version 4.0.0. Sie sortiert alle Zeichen der thailändischen Sprache in der Reihenfolge des Royal Thai Dictionary.
- **X'8003'** – Die Sortierfolge lautet UCA400_LSK und implementiert den UCA-Algorithmus (Unicode Collation Algorithm) auf der Grundlage von Unicode Standard Version 4.0.0. Sie sortiert alle Zeichen der slowakischen Sprache in der richtigen Reihenfolge.

Anmerkung:

- Bei einer sprachsensitiven Sortierfolge enthalten die ersten 256 Byte die Zeichenfolgedarstellung des Namens der Sortierfolge.
-

  Wichtig: Sortierungen auf der Basis des Unicode-Sortierungsalgorithmus der Unicode-Standardversion 4.0.0 gelten in Version 10.1 als veraltet und werden möglicherweise in einem zukünftigen Release entfernt. Weitere Informationen hierzu finden Sie im Abschnitt „Sortierungen auf der Basis des Unicode-Sortierungsalgorithmus der Unicode-Standardversion 4.0.0 gelten als veraltet" in *Neuerungen in DB2 Version 10.1*.

Wenn Sie diese internen Informationen zur Art der Sortierfolge verwenden, müssen Sie eine Bytefolgeumkehrung in Betracht ziehen, wenn Informationen zu einer Datenbank auf einer anderen Plattform abgerufen werden.

Sie können die Sortierfolge bei der Erstellung der Datenbank angeben.

connect_proc - Name der Verbindungsprozedur (Datenbankkonfigurationsparameter)

Dieser Datenbankkonfigurationsparameter ermöglicht Ihnen die Eingabe oder Aktualisierung eines zweiteiligen Namens für eine Verbindungsprozedur, die jedes Mal ausgeführt wird, wenn eine Anwendung eine Verbindung zur Datenbank herstellt.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Nach Member in einer DB2 pureScale-Umgebung konfigurierbar

Standardwert
`NULL`

Die folgenden Konventionen für Verbindungsprozeduren müssen befolgt werden, andernfalls wird ein Fehler zurückgegeben.

- Die Zeichenfolge, deren Länge nicht null sein darf, muss einen zweiteiligen Prozedurnamen angeben (also [schemaname].[prozedurname]).
- Der Verbindungsprozedurname (sowohl Schemaname als auch Prozedurname) darf nur die folgenden Zeichen enthalten:
  – A - Z

- a - z
- _ (Unterstreichungszeichen)
- 0 - 9

- Darüber hinaus müssen der Schemaname und der Prozedurname den Regeln für einen Standardbezeichner entsprechen.

Wenn für den Parameter **connect_proc** ein Wert mit einer Länge ungleich null konfiguriert ist, führt der Server die angegebene Prozedur implizit für jede neue Verbindung aus.

Hinweise zurVerwendung

- Für die Aktualisierung dieses Parameters ist eine Verbindung zur Datenbank erforderlich. Für die Aktualisierung dieses Parameters ist allerdings keine Verbindung zur Datenbank erforderlich, wenn die Datenbank inaktiviert ist.
- Der Parameter **connect_proc** kann nur mit der Option **IMMEDIATE** des Befehls **UPDATE DATABASE CONFIGURATION** definiert werden. Die Option **DEFERRED** kann beim Definieren des Parameters **connect_proc** nicht verwendet werden.
- Nur eine Prozedur mit genau null Parametern kann als Verbindungsprozedur verwendet werden. Wenn der Parameter **connect_proc** gesetzt ist, darf keine andere Prozedur mit demselben zweiteiligen Namen in der Datenbank vorhanden sein.
- Der Verbindungsprozedurname muss in der Datenbank vorhanden sein, bevor der Parameter **connect_proc** aktualisiert werden kann. Der Befehl **UPDATE DATABASE CONFIGURATION** schlägt fehl und es wird ein Fehler zurückgegeben, wenn die Verbindungsprozedur mit null Parametern nicht in der Datenbank vorhanden ist oder wenn mehrere Prozeduren mit demselben Namen vorhanden sind.
- Verwenden Sie in einer auf Datenbasis partitionierten Umgebung dieselbe Verbindungsprozedur für alle Partitionen.

country/region - Gebietscode der Datenbank

Dieser Parameter zeigt den *Gebietscode* an, der beim Erstellen der Datenbank verwendet wird.

Konfigurationstyp
Datenbank

Parametertyp
Informativ

database_consistent - Datenbank ist konsistent

Dieser Parameter gibt an, ob die Datenbank in einem konsistenten Zustand ist.

Konfigurationstyp
Datenbank

Parametertyp
Informativ

YES gibt an, dass alle Transaktionen mit COMMIT festgeschrieben oder mit ROLLBACK rückgängig gemacht wurden, sodass die Daten konsistent sind. Wenn es zu einem Systemabsturz kommt, während die Datenbank konsistent ist, sind keinerlei Maßnahmen erforderlich, um die Datenbank wieder verwendbar zu machen.

NO gibt an, dass noch eine Transaktion ansteht oder eine andere Funktion in der Datenbank noch nicht abgeschlossen wurde, sodass die Daten zu diesem Zeitpunkt nicht konsistent sind. Wenn es zu einem „Systemabsturz" kommt, während die Datenbank nicht konsistent ist, müssen Sie die Datenbank mit dem Befehl **RESTART DATABASE** erneut starten, um sie wieder verwendbar zu machen.

database_level - Release-Level der Datenbank

Dieser Parameter gibt den Release-Level des Datenbankmanagers an, der die Datenbank verwenden kann.

Konfigurationstyp
Datenbank

Parametertyp
Informativ

Im Fall eines nicht abgeschlossenen oder fehlgeschlagenen Upgrades gibt dieser Parameter den Release-Level der Datenbank vor dem Upgrade wieder, der sich vom Wert des Parameters **release** (Release-Level der Datenbankkonfigurationsdatei) unterscheiden kann. Ansonsten ist der Wert des Parameters **database_level** mit dem Wert des Parameters **release** identisch.

database_memory - Größe des gemeinsam genutzten Datenbankspeichers

Dieser Parameter gibt die Menge des Speichers an, die für den gemeinsam genutzten Speicherbereich einer Datenbank reserviert ist. Ist diese Menge geringer als die aus den einzelnen Speicherparametern berechnete Menge (beispielsweise aus Sperrenlisten, Zwischenspeichern von Dienstprogrammen, Pufferpools usw.) wird die größere Menge verwendet.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp

- Online konfigurierbar
- Nach Member in einer DB2 pureScale-Umgebung konfigurierbar

  Anmerkung: Wenn für diesen Parameter in einer DB2 pureScale-Umgebung andere Werte festgelegt werden, sollte **self_tuning_mem** nicht aktiviert werden.

Standardwert [Bereich]
Automatic [Computed, 0 - 4 294 967 295]

Maßeinheit
Seiten (4 KB)

Zuordnung
Wenn die Datenbank aktiviert wird

Freigabe
Wenn die Datenbank inaktiviert wird

Wenn dieser Parameter auf AUTOMATIC gesetzt wird, aktiviert dies die automatische Optimierung. Wenn sie aktiviert ist, bestimmt die Speicheroptimierungsfunktion den Gesamtspeicherbedarf für die Datenbank und erhöht oder verringert auf der Basis des aktuellen Bedarfs der Datenbank die Speichergröße, die für den gemeinsam genutzten Datenbankspeicher zugeordnet ist. Wenn zum Beispiel der aktuelle Bedarf der Datenbank hoch ist und ausreichend freier Speicher auf dem System zur Verfügung steht, wird mehr Speicher für den gemeinsam genutzten Datenbankspeicher in Anspruch genommen. Wenn der Bedarf an Datenbankspeicher sinkt oder die Größe des freien Speichers auf dem System auf einen zu niedrigen Wert zurückgeht, wird ein Teil des gemeinsam genutzten Datenbankspeichers freigegeben.

Die Speicheroptimierungsfunktion lässt je nach dem berechneten Vorteil einer Bereitstellung zusätzlichen Speichers für die Instanz immer eine minimale Menge an Speicher frei. Wenn eine Bereitstellung von mehr Speicher für eine Instanz einen großen Vorteil erzielt, hält die Speicheroptimierungsfunktion eine kleinere Größe an Speicher frei. Wenn der Vorteil kleiner ist, wird mehr freier Speicher behalten. Dadurch können Datenbanken in der Verteilung von Systemspeicher kooperieren.

Da die Speicheroptimierungsfunktion Speicherressourcen auf verschiedene Speicherkonsumenten verteilt, müssen mindestens zwei Speicherkonsumenten für die automatische Optimierung aktiviert sein, damit eine automatische Optimierung erfolgen kann.

Die automatische Optimierung dieses Konfigurationsparameters findet nur statt, wenn der Speicher mit automatischer Leistungsoptimierung für die Datenbank aktiviert ist (der Konfigurationsparameter **self_tuning_mem** muss auf ON gesetzt sein).

Um die Verwaltung dieses Parameters zu vereinfachen, wird der Datenbankmanager durch die Einstellung COMPUTED angewiesen, die erforderliche Speicherkapazität zu berechnen und den Speicher bei der Aktivierung der Datenbank zuzuordnen. Der Datenbankmanager ordnet außerdem zusätzlichen Speicher zu, um dem Speicherbedarf in Zeiten hoher Auslastung für die Zwischenspeicher im gemeinsam genutzten Datenbankspeicherbereich gerecht zu werden, wenn ein Zwischenspeicher seine konfigurierte Größe überschreitet. Andere Operationen, wie zum Beispiel dynamische Aktualisierungen der Konfiguration, haben ebenfalls Zugriff auf diesen zusätzlichen Speicher. Der Befehl **db2pd** mit der Option **-memsets** kann zur Überwachung des im gemeinsam genutzten Datenbankspeicherbereich verbliebenen nicht genutzten Speichers verwendet werden.

Empfehlung: Dieser Wert bleibt in der Regel bei der Einstellung AUTOMATIC. Für Umgebungen, die die Einstellung AUTOMATIC nicht unterstützten, sollte die Einstellung COMPUTED verwendet werden. Zum Beispiel kann der zusätzliche Speicher zur Erstellung neuer Pufferpools oder zur Vergrößerung vorhandener Pufferpools verwendet werden.

Anmerkung:

- Wenn Sie in Version 9.7 den Konfigurationsparameter **database_memory** auf AUTOMATIC setzen, ist die anfängliche gemeinsam genutzte Datenbankspeicherzuordnung die konfigurierte Größe aller für die Datenbank definierten Zwischenspeicher und Pufferpools, und der Speicher wird nach Bedarf vergrößert. Wenn der Parameter **database_memory** auf einen bestimmten Wert gesetzt wurde, wird die erforderliche Speicherkapazität von Beginn an während der Aktivierung der Datenbank zugeordnet. Wenn die anfängliche Speicherkapazität vom Betriebssystem nicht zugeordnet werden kann oder den Grenzwert des Parameters

instance_memory überschreitet, schlägt die Aktivierung der Datenbank mit dem folgenden SQL-Fehler fehl: SQL1084C Gemeinsam genutzte Speichersegmente können nicht zugeordnet werden.

- Wenn Sie in DB2 Version 9.7 auf einem Solaris-Betriebssystem den Parameter **database_memory** auf AUTOMATIC setzen, verwendet der Datenbankmanager umlagerbaren Speicher für den gemeinsam genutzten Datenbankspeicher. Bei Solaris-Betriebssystemen unter UltraSPARC versucht der Datenbankmanager, 64-KB-Speicherseiten zu verwenden, falls diese verfügbar sind. Sind keine 64-KB-Speicherseiten verfügbar, verwendet der Datenbankmanager 8-KB-Speicherseiten. Bei Solaris-Betriebssystemen auf Sun-x64-Sytemen verwendet der Datenbankmanager 4-KB-Speicherseiten. Durch die Verwendung kleinerer Speicherseiten kann es zu Leistungseinbußen kommen. Darüber hinaus besteht ein höherer Bedarf an Auslagerungsspeicher (in der Größe des gemeinsam genutzten Datenbankspeichers) aufgrund der Nutzung von umlagerbarem gemeinsam genutzten Speicher.
- Wenn Sie in DB2 Version 9.7 unter Solaris den Parameter **database_memory** auf den Wert COMPUTED oder einen numerischen Wert setzen, verwendet der Datenbankmanager ISM (Intimate Shared Memory) und große Seiten für den gemeinsam genutzten Datenbankspeicher.

Steuern der DB2-Speicherbelegung:

Wenn der Parameter **instance_memory** auf AUTOMATIC gesetzt wurde, wird beim Starten der Instanz (**db2start**) eine feste Obergrenze für die gesamte Speicherbelegung durch die Instanz festgelegt. Die tatsächliche Speicherbelegung durch den Datenbankmanager variiert je nach Auslastung. Wenn der Speichermanager für automatische Leistungsoptimierung (STMM) für die Optimierung des Datenbankspeichers (Parameter **database_memory**) aktiviert wurde (standardmäßig für neue Datenbanken der Fall), aktualisiert er während der Laufzeit dynamisch die Größe der für die Leistung kritischen Zwischenspeicher innerhalb des gemeinsam genutzten Speichers der Datenbank. Dies geschieht in Abhängigkeit vom freien physischen Hauptspeicher im System, während gleichzeitig sichergestellt wird, dass ausreichend freier Instanzspeicher (Parameter **instance_memory**) für funktionellen Speicherbedarf zur Verfügung steht. Weitere Informationen finden Sie im Abschnitt zum Konfigurationsparameter **instance_memory**.

Einschränkungen in einigen Linux[1]-Kerneln:

Aufgrund von Betriebssystemeinschränkungen in einigen Linux-Kerneln lässt der Speichermanager für eine automatische Leistungsoptimierung derzeit die Einstellung AUTOMATIC für den Parameter **database_memory** nicht zu. Jedoch ist diese Einstellung jetzt für diese Kernel unter der Voraussetzung zulässig, dass **instance_memory** auf einen bestimmten Wert gesetzt wird, nicht jedoch auf AUTOMATIC. Wenn **database_memory** auf AUTOMATIC gesetzt wurde und **instance_memory** später zurück auf AUTOMATIC gesetzt wird, wird der Konfigurationsparameter **database_memory** bei der nächsten Aktivierung der Datenbank automatisch mit der Einstellung COMPUTED aktualisiert. Wenn einige Datenbanken bereits aktiv sind, stoppt der Speichermanager für die automatische Leistungsoptimierung die Optimierung der gesamten Datenbankspeicherkapazitäten (Parameter **database_memory**).

[1]Unter Linux unterstützt dieser Parameter die Einstellung AUTOMATIC für RHEL5 und SUSE 10 SP1 und höher, unabhängig von der Einstellung für den Parameter **instance_memory**. Alle anderen validierten Linux-Varianten kehren zu der Einstellung COMPUTED zurück, wenn der Kernel diese Funktion nicht unterstützt.

dbheap - Zwischenspeicher für Datenbank

Dieser Parameter definiert die maximale Speicherkapazität, die vom Zwischenspeicher für die Datenbank verwendet werden kann.

Ab Version 9.5 hat dieser Datenbankkonfigurationsparameter den Standardwert AUTOMATIC, was bedeutet, dass der Datenbankzwischenspeicher nach Bedarf erhöht wird, bis entweder der Grenzwert des Parameters **database_memory** oder des Parameters **instance_memory** erreicht ist.

Konfigurationstyp
Datenbank

Parametertyp

- Online konfigurierbar
- Nach Member in einer DB2 pureScale-Umgebung konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
Automatic [32 - 524 288]

Anmerkung: Der Standardwert unterliegt nach der ursprünglichen Datenbankerstellung Änderungen durch den DB2 Configuration Advisor.

Maßeinheit
Seiten (4 KB)

Zuordnung
Wenn die Datenbank aktiviert wird

Freigabe
Wenn die Datenbank inaktiviert wird

Für jede Datenbank gibt es einen Datenbankzwischenspeicher, der vom Datenbankmanager für alle Anwendungen, die auf die Datenbank zugreifen, verwendet wird. Er enthält Steuerblockdaten für Tabellen, Indizes, Tabellenbereiche und Pufferpools. Er enthält außerdem Speicherbereich für den Protokollpuffer (**logbufsz**) sowie temporären Speicher, der von Dienstprogrammen verwendet wird. Daher ist die Größe des Zwischenspeichers von vielen Variablen abhängig. Die Steuerblockdaten werden im Zwischenspeicher gehalten, bis alle Anwendungen die Verbindung zur Datenbank getrennt haben.

Der Mindestspeicherbereich, den der Datenbankmanager zu Beginn benötigt, wird beim Herstellen der ersten Verbindung zugeordnet. Der Datenbereich wird nach Bedarf erweitert, bis entweder die konfigurierte Obergrenze erreicht ist oder, wenn die Einstellung AUTOMATIC gewählt wurde, bis der Speicher für den Parameter **database_memory** oder **instance_memory** oder für beide ausgeschöpft ist.

Als grobe Orientierung können Sie die folgende Formel verwenden, wenn Sie dem Konfigurationsparameter **dbheap** einen Wert zuordnen möchten:

```
10 KB pro Tabellenbereich + 4 KB pro Tabelle + (1 KB + 4*verwendete Speicherbereiche)
  pro Bereichsclustertabelle (RCT - Range-Clustered Table)
```

Der von Ihnen konfigurierte Wert **dbheap** steht nur für einen Teil des zugeordneten Datenbankzwischenspeicher. Der Datenbankzwischenspeicher ist der Hauptspeicherbereich, über den der globale Datenbankspeicherbedarf befriedigt wird. Die Größe ist so berechnet, dass zusätzlich zum Wert **dbheap** auch grundlegende

Zuordnungen für den Start der Datenbank enthalten sind. Tools mit Speicherbelegung wie der Memory Tracker, Snapshot Monitor und **db2pd** berichten über die statistischen Daten des größeren Datenbankzwischenspeichers. Es findet keine separate Verfolgung der Zuordnungen statt, die durch den Konfigurationsparameter **dbheap** repräsentiert werden. Daher ist es normal, dass die von den Tools dokumentierten statistischen Daten zur Speicherbelegung durch den Datenbankzwischenspeicher den für den Parameter **dbheap** konfigurierten Wert überschreiten.

Mithilfe des Datenbanksystemmonitors können Sie unter Verwendung des Elements **db_heap_top** (Maximal zugeordneter Datenbankzwischenspeicher) die größte Speichermenge ermitteln, die für den Datenbankzwischenspeicher verwendet wurde.

Anmerkung:

- Arbeitsklassensets und Arbeitsaktionssets von Workload Management (WLM) werden im Datenbankzwischenspeicher gespeichert. Jedoch wird dafür nur ein sehr kleiner Teil des Speichers benötigt.
- Informationen zu gesicherten Kontexten, zum Workload-Management sowie zu Prüfrichtlinien werden zwecks schneller Verarbeitung im Hauptspeicher zwischengespeichert. Dieser Speicher wird aus dem Zwischenspeicher für die Datenbank zugeordnet. Daher stellen benutzerdefinierte Objekte für gesicherte Kontexte, für das Workload-Management und für Prüfrichtlinien größere Speicheranforderungen an den Datenbankzwischenspeicher. In diesem Fall wird empfohlen, den Konfigurationsparameter für den Datenbankzwischenspeicher auf den Wert `AUTOMATIC` zu setzen, damit der Datenbankmanager die Größe des Zwischenspeichers für die Datenbank automatisch verwaltet.

db_mem_thresh - Schwellenwert für Datenbankspeicher

Dieser Parameter gibt den maximalen Prozentsatz des festgeschriebenen, jedoch momentan ungenutzten gemeinsamen Datenbankspeichers an, der vom Datenbankmanager zugelassen wird, bevor damit begonnen wird, festgeschriebene Speicherseiten freizugeben und an das Betriebssystem zurückzugeben.

Konfigurationstyp
Datenbank

Parametertyp

- Online konfigurierbar
- Nach Member in einer DB2 pureScale-Umgebung konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
`10 [0–100 ]`

Maßeinheit
Prozent

Dieser Datenbankkonfigurationsparameter bezieht sich darauf, wie der Datenbankmanager überschüssigen gemeinsam genutzten Speicher der Datenbank behandelt. Wenn Seiten von Speicher durch einen Prozess beansprucht werden, werden sie in der Regel für den Prozess fest reserviert. Das heißt, eine Speicherseite wird vom Betriebssystem zugeordnet und belegt einen Bereich entweder im physischen Speicher oder in einer Auslagerungsdatei auf der Platte. Abhängig von der Auslastung der Datenbank kann es zu gewissen Tageszeiten zu Spitzenbelastungen der Datenbank mit gemeinsam genutztem Speicher kommen. Wenn das Betriebssystem ein-

mal genügend Speicher für diese Spitzenlastzeiten reserviert hat, bleibt dieser Speicher reserviert, auch wenn der Speicherbedarf später wieder nachlässt.

Gültige Werte sind ganze Zahlen aus dem Bereich 0 (sofortige Freigabe jedes nicht genutzten gemeinsamen Datenbankspeichers) bis 100 (keine Freigabe von ungenutztem gemeinsamem Datenbankspeicher). Der Standardwert ist 10 (Freigabe von ungenutztem Speicher erst, wenn mehr als 10 % des gemeinsam genutzten Datenbankspeichers zurzeit nicht genutzt werden). Dieser Wert sollte für die meisten Auslastungen geeignet sein.

Dieser Konfigurationsparameter kann dynamisch aktualisiert werden. Dieser Parameter muss mit Vorsicht aktualisiert werden, da ein zu niedrig eingestellter Wert zur einer übermäßigen Speicherbelastung für das System führen kann (da ständig Speicherseiten reserviert und wieder freigegeben werden), während ein zu hoch eingestellter Wert möglicherweise verhindert, dass der Datenbankmanager überhaupt gemeinsam genutzten Datenbankspeicher an das Betriebssystem zur Verwendung durch andere Prozesse zurückgibt.

Dieser Konfigurationsparameter wird ignoriert (d. h. ungenutzte Seiten des gemeinsam genutzten Datenbankspeichers bleiben reserviert), wenn der Speicherbereich des gemeinsam genutzten Datenbankspeichers durch die Registrierdatenbankvariable **DB2_PINNED_BP** fixiert ist, für große Seiten durch die Registrierdatenbankvariable **DB2_LARGE_PAGE_MEM** konfiguriert ist oder die Freigabe von Speicher durch die Registrierdatenbankvariable **DB2MEMDISCLAIM** explizit inaktiviert ist.

Einige Versionen von Linux unterstützen die Freigabe von Unterbereichen eines gemeinsam genutzten Speichersegments an das Betriebssystem nicht. Auf solchen Plattformen wird dieser Parameter ignoriert.

date_compat - DATE-Kompatibilität (Datenbankkonfigurationsparameter)

Dieser Parameter gibt an, ob die DATE-Kompatibilitätssemantik, die mit dem Datentyp TIMESTAMP(0) verknüpft ist, auf die verbundene Datenbank angewendet wird.

Konfigurationstyp
Datenbank

Parametertyp
Informativ

Der Wert wird bei der Erstellung der Datenbank festgelegt und basiert auf der Einstellung der Registrierdatenbankvariablen **DB2_COMPATIBILITY_VECTOR** für den Datentyp DATE. Der Wert kann nicht geändert werden.

dec_to_char_fmt - Funktion zur Konvertierung von Dezimalwerten in Zeichenwerte (Konfigurationsparameter)

Dieser Parameter dient zur Steuerung des Ergebnisses der Skalarfunktion CHAR und der CAST-Spezifikation bei der Konvertierung von Dezimalwerten in Zeichenwerte.

Konfigurationstyp
Datenbank

Parametertyp

- Online konfigurierbar
- Nach Member in einer DB2 pureScale-Umgebung konfigurierbar

Siehe Auswirkungen einer Änderung des Werts von 'dec_to_char_fmt'.

Standardwert [Bereich]
NEW [NEW, V95]

Die Einstellung des Parameters bestimmt, ob führende Nullen und ein abschließendes Dezimalzeichen in das Ergebnis der Funktion CHAR eingefügt werden. Wenn Sie den Parameter auf den Wert `NEW` setzen, werden führende Nullen und abschließend Dezimalzeichen nicht mit eingefügt. Wenn Sie den Parameter auf den Wert `V95` setzen, werden führende Nullen und abschließende Dezimalzeichen eingefügt.

Führende Nullen und ein abschließendes Dezimalzeichen werden auch in das Ergebnis der Skalarfunktion CHAR_OLD eingefügt, die dieselbe Syntax wie die Funktion CHAR besitzt.

Beim Upgrade von Datenbanken, die vor Version 9.7 erstellt und dann auf Version 9.7 oder höher aktualisiert wurden, wird der Parameter **dec_to_char_fmt** standardmäßig auf *V95* eingestellt.

Auswirkungen einer Änderung des Werts von 'dec_to_char_fmt'

- MQTs (Materialized Query Tables), die Sie vor Version 9.7 erstellt haben, können Ergebnisse enthalten, die von den Ergebnissen der MQTs abweichen, die Sie unter Verwendung der Einstellung `NEW` erstellt haben. Um sicherzustellen, dass zuvor erstellte MQTs nur Daten enthalten, die dem neuen Format entsprechen, aktualisieren Sie diese MQTs mithilfe der Anweisung REFRESH TABLE.
- Die Ergebnisse eines Triggers können durch das geänderte Format beeinflusst werden. Ein Einstellen des Parameters auf den Wert `NEW` zum Ändern des Formats hat keine Auswirkung auf Daten, die bereits geschrieben wurden.
- Integritätsbedingungen, die es zuließen, dass Daten in eine Tabelle eingefügt wurden, könnten nach einer erneuten Prüfung dieselben Daten nun zurückweisen. Ebenso könnten Integritätsbedingungen, die eine Einfügung von Daten in eine Tabelle bisher nicht zuließen, nach der erneuten Prüfung dieselben Daten nun akzeptieren. Verwenden Sie die Anweisung SET INTEGRITY, um eine Prüfung auf Daten in einer Tabelle auszuführen, die eine Integritätsbedingung jetzt nicht mehr erfüllen, und diese Daten zu korrigieren.
- Kompilieren Sie nach einer Änderung des Parameters **dec_to_char_fmt** alle statischen SQL-Pakete erneut, die vom Wert einer generierten Spalte abhängen, deren Ergebnisse von der Änderung am Wert des Parameters **dec_to_char_fmt** betroffen sind. Zur Ermittlung der betroffenen statischen SQL-Pakete müssen Sie alle Pakete kompilieren und mit dem Befehl **db2rbind** (Rebind) erneut binden.

decflt_rounding - Rundungsmodus für dezimale Gleitkommawerte (Konfigurationsparameter)

Mit diesem Parameter können Sie den Rundungsmodus für dezimale Gleitkommawerte (DECFLOAT) angeben. Der Rundungsmodus betrifft Operationen mit dezimalen Gleitkommawerten auf dem Server sowie den Ladevorgang (**LOAD**).

Konfigurationstyp
Datenbank

Parametertyp
Konfigurierbar

Siehe „Auswirkungen einer Änderung von decflt_rounding” auf Seite 870.

Standardwert [Bereich]
ROUND_HALF_EVEN [ROUND_CEILING, ROUND_FLOOR, ROUND_HALF_UP, ROUND_DOWN]

DB2-Datenbanksysteme unterstützen fünf IEEE-konforme Rundungsmodi für dezimale Gleitkommawerte. Der Rundungsmodus gibt an, wie das Ergebnis einer Berechnung zu runden ist, wenn das Ergebnis die zulässige Anzahl Stellen überschreitet. Die Rundungsmodi sind wie folgt definiert:

ROUND_CEILING
Aufrunden (gegen positiv Unendlich runden). Wenn alle verworfenen Stellen null sind oder wenn das Vorzeichen negativ ist, bleibt das Ergebnis unverändert. Andernfalls wird der Ergebniskoeffizient um 1 erhöht (aufgerundet).

ROUND_FLOOR
Abrunden (gegen negativ Unendlich runden). Wenn alle verworfenen Stellen null sind oder wenn das Vorzeichen positiv ist, bleibt das Ergebnis unverändert. Andernfalls bleibt das Vorzeichen negativ und der Ergebniskoeffizient wird um 1 erhöht.

ROUND_HALF_UP
Zum näherliegenden Wert der höheren Dezimalstelle auf- oder abrunden. Bei gleichem Abstand, um 1 aufrunden. Wenn die verworfenen Stellen einen Wert größer-gleich der Hälfte des Werts 1 (0,5) an der nächsthöheren Dezimalstelle darstellen, wird der Ergebniskoeffizient um 1 erhöht (aufrunden). Andernfalls werden die verworfenen Stellen (0,5 oder weniger) ignoriert.

ROUND_HALF_EVEN
Zum näherliegenden Wert der nächsthöheren Dezimalstelle auf- oder abrunden. Bei gleichem Abstand so runden, dass die letzte Ziffer gerade ist. Wenn die verworfenen Stellen einen Wert größer als die Hälfte des Werts 1 (0,5) an der nächsthöheren Dezimalstelle darstellen, wird der Ergebniskoeffizient um 1 erhöht (aufgerundet). Wenn sie einen Wert kleiner als 0,5 darstellen, wird der Ergebniskoeffizient nicht angepasst, d. h., die verworfenen Stellen werden ignoriert. Wenn sie genau die Hälfte (0,5) darstellen, bleibt der Ergebniskoeffizient unverändert, wenn die letzte rechte Ziffer gerade ist. Sie wird um 1 erhöht (aufgerundet), wenn die letzte rechte Ziffer ungerade ist, damit eine gerade Ziffer entsteht. Dieser Rundungsmodus ist laut Spezifikation für IEEE-konforme dezimale Gleitkommawerte der Standardrundungsmodus und wird auch in DB2-Datenbankprodukten als Standardrundungsmodus verwendet.

ROUND_DOWN
Abrunden (Abschneiden). Die verworfenen Stellen werden ignoriert.

Tabelle 137 zeigt die Ergebnisse des Auf-/Abrundens der Werte 12,341, 12,345, 12,349, 12,355 und -12,345 auf jeweils vier Stellen für die verschiedenen Rundungsmodi:

Tabelle 137. Rundungsmodi für dezimale Gleitkommawerte

Rundungsmodus	**12,341**	**12,345**	**12,349**	**12,355**	**-12,345**
ROUND_DOWN	12,34	12,34	12,34	12,35	-12,34
ROUND_HALF_UP	12,34	12,35	12,35	12,36	-12,35
ROUND_HALF_EVEN	12,34	12,34	12,35	12,36	-12,34

Tabelle 137. Rundungsmodi für dezimale Gleitkommawerte (Forts.)

Rundungsmodus	12,341	12,345	12,349	12,355	-12,345
ROUND_FLOOR	12,34	12,34	12,34	12,35	-12,35
ROUND_CEILING	12,35	12,35	12,35	12,36	-12,34

Auswirkungen einer Änderung von decflt_rounding

- MQTs, die früher erstellt wurden, könnten Ergebnisse enthalten, die sich von den Ergebnissen unterscheiden, die mit dem neuen Rundungsmodus erstellt werden. Um dieses Problem zu lösen, sollten Sie potenziell betroffene MQTs aktualisieren.
- Die Ergebnisse eines Triggers können durch den neuen Rundungsmodus beeinträchtigt werden. Eine Änderung hat keinerlei Auswirkungen auf die bereits geschriebenen Daten.
- Integritätsbedingungen, die das Einfügen von Daten in eine Tabelle ermöglichten, weisen (im Falle einer erneuten Auswertung) genau diese Daten möglicherweise zurück. Im Gegensatz dazu akzeptieren Integritätsbedingungen, die das Einfügen von Daten in eine Tabelle nicht ermöglichten, (im Falle einer erneuten Auswertung) genau diese Daten. Verwenden Sie die Anweisung SET INTEGRITY, um nach solchen Problemen zu suchen und sie zu beheben. Der Wert einer generierten Spalte, deren Berechnung von **decflt_rounding** abhängig ist, kann für zwei Zeilen, die abgesehen vom generierten Spaltenwert identisch sind, unterschiedlich sein, wenn die eine Zeile vor der Änderung an **decflt_rounding** und die andere nach der Änderung an **decflt_rounding** eingefügt wurde.
- Der Rundungsmodus wird nicht in Abschnitte hinein kompiliert. Aus diesem Grund muss statisches SQL nach der Änderung von **decflt_rounding** nicht erneut kompiliert werden.

Anmerkung: Der Wert dieses Konfigurationsparameters wird nicht dynamisch geändert, sondern wird erst wirksam, wenn alle Anwendungen ihre Verbindung zur Datenbank getrennt haben. Wenn die Datenbank aktiviert ist, muss sie inaktiviert werden.

dft_degree - Grad der Parallelität

Dieser Parameter gibt den Standardwert für das Sonderregister CURRENT DEGREE und die Bindeoption **DEGREE** an.

Konfigurationstyp
Datenbank

Parametertyp

- Online konfigurierbar
- Nach Member in einer DB2 pureScale-Umgebung konfigurierbar

Weitergabeklasse
Verbindung

Standardwert [Bereich]
1 [-1(ANY), 1 - 32 767]

Anmerkung: Der Standardwert unterliegt nach der ursprünglichen Datenbankerstellung Änderungen durch den DB2 Configuration Advisor.

Der Standardwert ist 1.

Bei Angabe des Werts 1 wird keine partitionsinterne Parallelität verwendet. Der Wert -1 (oder ANY) bedeutet, dass das Optimierungsprogramm den Grad der partitionsinternen Parallelität je nach Anzahl der Prozessoren und der Art der Abfrage festlegt.

Der Grad der partitionsinternen Parallelität für eine SQL-Anweisung wird während der Kompilierung der Anweisung mithilfe des Sonderregisters CURRENT DEGREE oder der Bindeoption **DEGREE** angegeben. Der maximale Grad der partitionsinternen Parallelität zur Laufzeit für eine aktive Anwendung wird mit dem Befehl **SET RUNTIME DEGREE** angegeben. Der Konfigurationsparameter 'Maximaler Grad der Parallelität bei Abfragen' (**max_querydegree**) gibt den maximalen Grad der partitionsinternen Parallelität für alle SQL-Abfragen an.

Der zur Laufzeit tatsächlich verwendete Parallelitätsgrad ist der niedrigste der folgenden Werte:

- Konfigurationsparameter **max_querydegree**
- Grad der Anwendung zur Laufzeit
- Parallelitätsgrad bei der Kompilierung der SQL-Anweisung

dft_extent_sz - Standardwert für EXTENTSIZE bei Tabellenbereichen

Mit diesem Parameter wird der Standardwert für EXTENTSIZE bei Tabellenbereichen festgelegt.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
32 [2 – 256]

Anmerkung: Der Standardwert unterliegt nach der ursprünglichen Datenbankerstellung Änderungen durch den DB2 Configuration Advisor.

Maßeinheit
Seiten

Bei der Erstellung eines Tabellenbereichs kann wahlfrei EXTENTSIZE n angegeben werden, wobei n die Speicherbereichsgröße ist. Wenn Sie EXTENTSIZE in der Anweisung CREATE TABLESPACE nicht angeben, verwendet der Datenbankmanager den Wert, der durch diesen Parameter definiert wird.

Empfehlung: In vielen Fällen geben Sie die Speicherbereichsgröße bei der Erstellung eines Tabellenbereichs explizit an. Bevor Sie einen Wert für diesen Parameter wählen, sollten Sie verstehen, wie die Speicherbereichsgröße in der Anweisung CREATE TABLESPACE explizit gewählt wird.

In einer DB2 pureScale-Umgebung sollten Sie für die Speicherbereichsgröße mindestens den Standardwert (32 Seiten) verwenden. Bei dieser minimalen Speicherbereichsgröße wird die Menge der internen Nachrichtenübertragungen in der DB2 pureScale-Umgebung verringert, wenn Speicherbereiche für eine Tabelle oder einen Index hinzugefügt werden. Hier einige Beispiele für Fälle, in denen häufig neue

Speicherbereiche zugeordnet werden und in denen größere Speicherbereiche von Vorteil sind: die Dienstprogramme LOAD und IMPORT, die Anweisung CREATE INDEX sowie Massendateneinfügungen aus Anwendungen.

dft_loadrec_ses - Standardanzahl von Sitzungen für Recovery

Mit diesem Parameter wird die Standardanzahl von Sitzungen angegeben, die während des Wiederabrufens einer Tabellenladekopie verwendet werden.

Konfigurationstyp
Datenbank

Parametertyp
- Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
1 [1 - 30 000]

Maßeinheit
Zähler

Der Wert für diesen Parameter sollte auf die Anzahl von E/A-Sitzungen gesetzt werden, die zum Wiederabrufen einer Ladekopie verwendet werden sollen. Das Wiederabrufen einer Ladekopie ist eine ähnliche Operation wie ein Restore. Sie können diesen Parameter durch Einträge überschreiben, die sich in der Datei mit den Angaben zur Speicherposition der exportierten Daten befinden. Diese Datei wird von der Umgebungsvariablen **DB2LOADREC** angegeben.

Die Standardanzahl der Puffer, die für den Abruf der Ladekopien verwendet werden, ist um zwei größer als der Wert dieses Parameters. Die Anzahl der Puffer kann ebenfalls in der Datei mit den Angaben zur Speicherposition der exportierten Daten überschrieben werden.

Dieser Parameter ist nur gültig, wenn die aktualisierende Recovery aktiviert ist.

dft_mttb_types - Standardtypen von verwalteten Tabellen zur Optimierung

Mit diesem Parameter wird der Standardwert für das Sonderregister CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION angegeben. Der Wert dieses Registers bestimmt, welche Typen mit REFRESH DEFERRED definierter MQTs der Abfrageoptimierung verwendet werden.

Konfigurationstyp
Datenbank

Parametertyp
Konfigurierbar

Standardwert [Bereich]
SYSTEM [ALL, NONE, FEDERATED_TOOL, SYSTEM, USER oder eine Liste von Werten]

Sie können eine Liste von Werten durch Kommata getrennt angeben. Zum Beispiel: 'USER,FEDERATED_TOOL'. ALL und NONE können nicht zusammen mit anderen Werten angegeben werden, und Sie können denselben Wert nur einmal angeben. Zur Verwendung in Verbindung mit den APIs **db2CfgSet** und **db2CfgGet** lauten die

akzeptierbaren Parameterwerte wie folgt: 8 (ALL), 4 (NONE), 16 (FEDERATED-_TOOL), 1 (SYSTEM) und 2 (USER). Es können mehrere Werte zusammen angegeben werden; dazu müssen Sie das bitweise ODER verwenden; Beispiel: 18 wäre das Äquivalent zu USER,FEDERATED_TOOL. Wie zuvor dürfen die Werte 4 und 8 nicht mit anderen Werten verwendet werden.

dft_prefetch_sz - Standardwert für PREFETCHSIZE

Mit diesem Parameter wird der Standardwert für PREFETCHSIZE bei Tabellenbereichen festgelegt.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
`Automatic [0 - 32 767]`

Anmerkung: Der Standardwert unterliegt nach der ursprünglichen Datenbankerstellung Änderungen durch den DB2 Configuration Advisor.

Maßeinheit
Seiten

Bei der Erstellung eines Tabellenbereichs kann PREFETCHSIZE optional mit dem Wert AUTOMATIC oder *n* angegeben werden, wobei *n* die Anzahl der Seiten ist, die der Datenbankmanager liest, wenn ein Vorablesezugriff erfolgt. Wenn Sie in der Anweisung CREATE TABLESPACE keinen Wert für PREFETCHSIZE angeben, verwendet der Datenbankmanager den aktuellen Wert des Parameters **dft_prefetch_sz**.

Wenn beim Erstellen eines Tabellenbereichs die Vorablesezugriffsgröße auf `AUTOMATIC` gesetzt ist, berechnet und aktualisiert DB2 den Wert für PREFETCHSIZE des Tabellenbereichs automatisch.

Diese Berechnung erfolgt zu folgenden Zeitpunkten:

- Wenn die Datenbank gestartet wird
- Bei der anfänglichen Erstellung eines Tabellenbereichs mit AUTOMATIC DFT_PREFETCH_SZ
- Bei einer Änderung der Anzahl von Containern für einen Tabellenbereich durch die Ausführung einer Anweisung ALTER TABLESPACE
- Bei einer Änderung des Werts für PREFETCHSIZE für einen Tabellenbereich in AUTOMATIC durch eine Anweisung ALTER TABLESPACE

Der Status `AUTOMATIC` für PREFETCHSIZE kann aktiviert oder inaktiviert werden, wenn der PREFETCHSIZE-Wert manuell durch eine Anweisung ALTER TABLESPACE aktualisiert wird.

Empfehlung: Mit Tools zur Systemüberwachung können Sie feststellen, ob Ihre CPU ausgelastet ist, während das System auf Ein-/Ausgaben wartet. Die Erhöhung dieses Parameterwerts kann nützlich sein, wenn für die Tabellenbereiche, die verwendet werden, kein Wert für PREFETCHSIZE definiert ist.

Dieser Parameter enthält den Standardwert für die gesamte Datenbank und ist eventuell nicht für alle Tabellenbereiche innerhalb der Datenbank geeignet. Beispielsweise kann der Wert 32 für einen Tabellenbereich mit dem Wert 32 Seiten für EXTENTSIZE geeignet sein, aber nicht für einen Tabellenbereich mit dem Wert 25 Seiten für EXTENTSIZE. Im Idealfall sollten Sie den Wert für PREFETCHSIZE für jeden Tabellenbereich explizit angegeben.

Sie sollten als Wert für diesen Parameter einen Faktor oder ein ganzzahliges Vielfaches des Wertes des Parameters **dft_extent_sz** angeben, um die E/A-Operationen für Tabellenbereiche zu minimieren, die mit dem Standardwert für EXTENTSIZE (Parameter **dft_extent_sz**) definiert sind. Wenn z. B. für den Parameter **dft_extent_sz** der Wert 32 angegeben ist, könnten Sie den Wert von **dft_prefetch_sz** auf 16 (einen Bruchteil von 32) oder auf 64 (ein ganzzahliges Vielfaches von 32) setzen. Wenn für PREFETCHSIZE ein Vielfaches des Werts für EXTENTSIZE angegeben ist, kann der Datenbankmanager parallele E/A-Operationen ausführen, wenn die folgenden Bedingungen erfüllt sind:

- Die Bereiche, die vorab gelesen werden, befinden sich auf verschiedenen physischen Einheiten.
- Es sind mehrere E/A-Server konfiguriert (**num_ioservers**).

dft_queryopt - Standardabfrageoptimierungsklasse

Mit der Abfrageoptimierungsklasse können Sie das Optimierungsprogramm anweisen, beim Kompilieren von SQL- und XQuery-Abfragen verschiedene Grade der Optimierung zu verwenden. Dieser Parameter bietet zusätzliche Flexibilität, indem die Standardabfrageoptimierungsklasse für den Fall festgelegt wird, dass weder die Anweisung SET CURRENT QUERY OPTIMIZATION noch die Option **QUERYOPT** mit dem Befehl **BIND** verwendet wird.

Konfigurationstyp
Datenbank

Parametertyp

- Online konfigurierbar
- Nach Member in einer DB2 pureScale-Umgebung konfigurierbar

Weitergabeklasse
Verbindung

Standardwert [Bereich]
5 [0 — 9]

Anmerkung: Der Standardwert unterliegt nach der ursprünglichen Datenbankerstellung Änderungen durch den DB2 Configuration Advisor.

Maßeinheit
Abfrageoptimierungsklasse (siehe nachstehende Liste)

Derzeit sind folgende Abfrageoptimierungsklassen definiert:

- 0 - Minimale Abfrageoptimierung
- 1 - In etwa vergleichbar mit DB2 Version 1
- 2 - Geringe Optimierung
- 3 - Mittlere Abfrageoptimierung
- 5 - Starke Abfrageoptimierung, die über heuristische Methoden den Aufwand bei der Auswahl eines Zugriffsplans reduziert (dies ist der Standardwert)
- 7 - Starke Abfrageoptimierung

- 9 - Maximale Abfrageoptimierung

dft_refresh_age - Standardaktualisierungsalter

Dieser Parameter stellt die maximale Dauer seit der Verarbeitung einer Anweisung REFRESH TABLE für eine bestimmte MQT (Materialized Query Table, gespeicherte Abfragetabelle) dar. Nach der Überschreitung dieses Zeitlimits wird die MQT erst zur Erfüllung von Abfragen verwendet, wenn die MQT aktualisiert ist.

Konfigurationstyp
Datenbank

Parametertyp
Konfigurierbar

Standardwert [Bereich]
0 [0, 99999999999999 (ANY)]

Maßeinheit
Sekunden

Für diesen Parameter wird der für REFRESH AGE verwendete Standardwert eingesetzt, wenn das Sonderregister CURRENT REFRESH AGE nicht angegeben wird. Dieser Parameter gibt einen Zeitmarkendifferenzwert mit dem Datentyp DECIMAL(20,6) an. Wenn für CURRENT REFRESH AGE der Wert 99999999999999 (ANY) gilt und die Abfrageoptimierungsklasse (QUERY OPTIMIZATION) den Wert zwei (oder fünf und höher) hat, werden MQTs vom Typ REFRESH DEFERRED zum Optimieren der Verarbeitung einer dynamischen Abfrage herangezogen.

dft_schemas_dcc - Standarddatenerfassung für neue Schemas (Konfigurationsparameter)

Dieser Parameter ermöglicht die Steuerung der Standardeinstellung für DATA CAPTURE CHANGES bei neu erstellten Schemas für Replikationszwecke.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
`No [Yes; No ]`

Standardmäßig wird der Parameter **dft_schemas_dcc** auf den Wert `NO` gesetzt. Wenn die Einstellung `YES` angegeben ist, weisen alle neu erstellten Schemas standardmäßig die Klausel DATA CAPTURE CHANGES auf.

dft_sqlmathwarn - Bei arithmetischen Ausnahmebedingungen fortsetzen

Mit diesem Parameter wird der Standardwert festgelegt, der bestimmt, ob arithmetische Fehler und Fehler bei Abfrageumwandlungen beim Ausführen von SQL-Anweisungen als Fehler oder als Warnungen behandelt werden.

Konfigurationstyp
Datenbank

Parametertyp
Konfigurierbar

Standardwert [Bereich]
No [No, Yes]

Bei statischen SQL-Anweisungen wird der Wert dieses Parameters dem Paket während des Bindens zugeordnet. Bei dynamischen SQL-DML-Anweisungen wird der Wert dieses Parameters verwendet, wenn die Anweisung vorbereitet wird (PREPARE).

Achtung: Wenn Sie den Wert des Parameters **dft_sqlmathwarn** für eine Datenbank ändern, kann sich das Verhalten von Prüfungen auf Integritätsbedingung, Triggern und Sichten, die arithmetische Ausdrücke enthalten, ändern. Dies wiederum kann sich auf die Datenintegrität der Datenbank auswirken. Sie sollten die Einstellung des Parameters **dft_sqlmathwarn** für eine Datenbank nur nach einer gründlichen Analyse der Auswirkungen der neuen Behandlung arithmetischer Ausnahmebedingungen auf Integritätsprüfungen, Trigger und Sichten ändern. Auch alle weiteren Änderungen sind mit derselben Sorgfalt vorzunehmen.

Betrachten Sie beispielsweise die folgende Prüfung auf Integritätsbedingung, die eine arithmetische Divisionsoperation enthält:

```
A/B > 0
```

Wenn **dft_sqlmathwarn** gleich No ist und eine Einfügeoperation (INSERT) mit B=0 versucht wird, wird die Division durch 0 als arithmetischer Fehler verarbeitet. Die Einfügeoperation schlägt fehl, weil der DB2-Datenbankmanager die Integritätsbedingung nicht prüfen kann. Wenn der Parameter **dft_sqlmathwarn** auf Yes gesetzt wird, wird die Division durch 0 als arithmetische Warnung mit dem Ergebnis NULL verarbeitet. Das Ergebnis NULL führt dazu, dass das Vergleichselement mit dem Wert UNKNOWN ausgewertet wird und die Einfügeoperation erfolgreich ist. Wenn **dft_sqlmathwarn** wieder in No geändert wird, schlägt der Versuch, dieselbe Zeile einzufügen, fehl, weil der Fehler der Division durch 0 den DB2-Datenbankmanager daran hindert, die Integritätsbedingung auszuwerten. Die Zeile, die mit B=0 eingefügt wurde, als der Parameter **dft_sqlmathwarn** den Wert Yes hatte, bleibt in der Tabelle und kann ausgewählt werden. Aktualisierungen der Zeile, für die die Integritätsbedingung ausgewertet wird, schlagen fehl, während Aktualisierungen, die keine erneute Prüfung der Integritätsbedingung erfordern, erfolgreich ausgeführt werden.

Bevor Sie den Parameter **dft_sqlmathwarn** von No in Yes ändern, sollten Sie in Betracht ziehen, die Integritätsbedingung so umzuschreiben, dass sie Nullwerte aus arithmetischen Ausdrücken gesondert behandelt. Beispiel:

```
( A/B > 0 ) AND ( CASE
                    WHEN A IS NULL THEN 1
                    WHEN B IS NULL THEN 1
                    WHEN A/B IS NULL THEN 0
                    ELSE 1
                    END
                  = 1 )
```

Diese Bedingung kann verwendet werden, wenn A und B Nullwerte haben können, d. h. die Dateneingabe optional ist. Wenn A oder B keinen Nullwert haben kann, kann die entsprechende Klausel WHEN...IS NULL entfernt werden.

Bevor Sie den Parameter **dft_sqlmathwarn** von Yes in No ändern, sollten Sie zunächst prüfen, welche Daten inkonsistent werden könnten, indem Sie zum Beispiel Vergleichselemente wie die folgenden verwenden:

```
WHERE A IS NOT NULL AND B IS NOT NULL AND A/B IS NULL
```

Wenn inkonsistente Zeilen isoliert sind, können Sie geeignete Maßnahmen zur Korrektur der Inkonsistenz ergreifen, bevor Sie den Parameter **dft_sqlmathwarn** ändern. Sie können Integritätsbedingungen mit arithmetischen Ausdrücken nach der Änderung auch manuell gegenprüfen. Dazu setzen Sie die betroffenen Tabellen in den Status *Überprüfung anstehend* (mit der Klausel OFF der Anweisung SET CONSTRAINTS) und fordern anschließend eine Prüfung an (mit der Klausel IMMEDIATE CHECKED der Anweisung SET CONSTRAINTS). Inkonsistente Daten werden durch einen arithmetischen Fehler angezeigt, der verhindert, dass die Integritätsbedingung ausgewertet wird.

Empfehlung: Verwenden Sie die Standardeinstellung no, sofern Sie keine Abfragen benötigen, deren Verarbeitung arithmetische Ausnahmebedingungen einschließt. In diesem Fall geben Sie den Wert yes an. Dieser Fall kann eintreten, wenn Sie SQL-Anweisungen verarbeiten, die auf anderen Datenbankmanagern unabhängig von möglichen arithmetischen Ausnahmebedingungen Ergebnisse liefern.

discover_db - Discovery-Unterstützung für diese Datenbank

Mit diesem Parameter kann verhindert werden, dass Informationen zu einer Datenbank an einen Client zurückgegeben werden, wenn eine Discovery-Anforderung auf dem Server empfangen wird.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
ENABLE [DISABLE, ENABLE]

Standardmäßig ist dieser Parameter so eingestellt, dass die Discovery-Unterstützung für diese Datenbank aktiviert ist.

Durch Ändern dieses Parameterwerts in Disable können Datenbanken, die sensible Daten enthalten, vor dem Discovery-Prozess verdeckt werden. Diese Maßnahme kann zusätzlich zu anderen Datenbanksicherheitsfunktionen für die Datenbank ergriffen werden.

dlchktime - Zeitintervall für Prüfung von Deadlocks

Dieser Parameter definiert die Häufigkeit, mit der alle mit der Datenbank verbundenen Anwendungen vom Datenbankmanager auf Deadlocks überprüft werden.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
10 000 (10 Sekunden) [1 000 - 600 000]

Maßeinheit
Millisekunden

Beim Auftreten von Deadlocks müssen zwei oder mehr Anwendungen, die auf dieselbe Datenbank zugreifen, für einen unbegrenzten Zeitraum auf eine Ressource warten. Die Anwendungen warten unendlich lange, da jede Anwendung eine Ressource gesperrt hat, die von der anderen Anwendung zur Fortsetzung der Verarbeitung benötigt wird.

Anmerkung:

1. In einer Umgebung mit partitionierten Datenbanken gilt dieser Parameter nur für den Katalogknoten.
2. In einer Umgebung mit partitionierten Datenbanken wird ein Deadlock erst nach der zweiten Iteration markiert.

Empfehlung: Durch Erhöhung des Werts für diesen Parameter wird die Häufigkeit der Prüfung auf Deadlocks verringert, wodurch die Zeit, die Anwendungsprogramme auf die Aufhebung von Deadlocks warten müssen, erhöht wird.

Durch Angabe eines niedrigeren Werts für diesen Parameter wird die Häufigkeit der Prüfung auf Deadlocks erhöht, wodurch die Zeit, die Anwendungsprogramme auf die Aufhebung von Deadlocks warten müssen, verkürzt, die Zeit, die der Datenbankmanager für die Prüfung auf Deadlocks verwendet, jedoch verlängert wird. Wenn das Prüfintervall zu klein ist, kann dies zu einer Verschlechterung der Laufzeitleistung führen, weil der Datenbankmanager häufig nach Deadlocks sucht. Wenn dieser Parameter mit einem niedrigeren Wert versehen wird, um den gemeinsamen Zugriff zu verbessern, sollten Sie darauf achten, dass die Parameter **maxlocks** und **locklist** entsprechend eingestellt sind, um unnötige Sperreneskalationen zu vermeiden, die zu Zugriffskonflikten und weiteren Deadlocks führen können.

enable_xmlchar - Ermöglichen der Konvertierung in XML (Konfigurationsparameter)

Dieser Parameter legt fest, ob für Nicht-BIT DATA CHAR-Ausdrücke (bzw. CHAR-Ausdrücke) in einer SQL-Anweisung XMLPARSE-Operationen durchgeführt werden können.

Konfigurationstyp
Datenbank

Parametertyp
Konfigurierbar

Standardwert [Bereich]
Yes [Yes; No]

Wenn pureXML-Funktionen in einer Nicht-Unicode-Datenbank verwendet werden, kann es durch die XMLPARSE-Funktion zu Zeichensubstitutionen kommen, da SQL-Zeichenfolgedaten von der Client-Codepage in die Datenbankcodepage und anschließend in die Unicode-Codepage zur internen Speicherung umgesetzt werden. Wenn **enable_xmlchar** auf NO gesetzt wird, wird die Verwendung von Zeichendatentypen bei der XML-Syntaxanalyse blockiert, und bei sämtlichen Versuchen, Zeichentypen in eine Nicht-Unicode-Datenbank einzufügen, wird ein Fehler generiert. Der BLOB-Datentyp und FOR BIT DATA-Datentypen sind weiterhin zulässig,

wenn **enable_xmlchar** auf NO gesetzt wird, da keine Codepagekonvertierung erfolgt, wenn diese Datentypen zum Einfügen von XML-Daten in eine Datenbank verwendet werden.

Standardmäßig ist **enable_xmlchar** auf YES gesetzt, sodass die Syntaxanalyse von Zeichendatentypen möglich ist. In diesem Fall müssen Sie sicherstellen, dass sämtliche XML-Daten, die eingefügt werden sollen, ausschließlich Codepunkte enthalten, die Teil der Datenbankcodepage sind, damit keine Substitutionszeichen während des Einfügevorgangs der XML-Daten eingeführt werden.

Anmerkung: Der Client muss die Verbindung zum Agenten unterbrechen und anschließend die Verbindung wiederherstellen, damit diese Änderung wirksam wird.

failarchpath - Protokollarchivpfad für Funktionsübernahme

Dieser Parameter gibt einen Pfad an, in dem das DB2-Datenbanksystem versucht, Protokolldateien zu archivieren, wenn die Protokolldateien weder am primären Archivierungsziel noch am sekundären (falls definiert) archiviert werden können, weil diese Ziele von einem Datenträgerproblem betroffen sind. Der angegebene Pfad muss auf eine Platte verweisen.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
Null [ ]

Wenn sich Protokolldateien in dem durch den aktuellen Wert von **failarchpath** angegebenen Pfad befinden, werden sämtliche Aktualisierungen für **failarchpath** nicht sofort wirksam. Die Aktualisierung wird dann wirksam, wenn die Verbindung zu allen Anwendungen getrennt ist.

groupheap_ratio - Für Anwendungsgruppenzwischenspeicher vorgesehener Speicher in Prozent

Dieser Parameter ist in Version 9.5 veraltet, wird jedoch von Datenservern und Clients mit Versionen vor Version 9.5 weiterhin verwendet. Jeder für diesen Konfigurationsparameter angegebene Wert wird vom Datenbankmanager in DB2 Version 9.5 ignoriert. In Version 9.5 wurde er durch den Konfigurationsparameter **appl_memory** ersetzt.

Anmerkung: Die nachfolgenden Informationen gelten nur für Datenserver und Clients einer Version vor Version 9.5.

Dieser Parameter gibt den Prozentsatz an Speicher im gemeinsam genutzten Speicher zur Anwendungssteuerung an, der für den von der Anwendungsgruppe gemeinsam genutzten Zwischenspeicher vorgesehen ist.

Konfigurationstyp
Datenbank

Parametertyp
Konfigurierbar

Standardwert [Bereich]
`70 [1 – 99 ]`

Maßeinheit
Prozent

Dieser Parameter hat keine Auswirkungen auf eine nicht partitionierte Datenbank mit inaktiviertem Konzentrator und inaktivierter partitionsinterner Parallelität.

Empfehlung: Behalten Sie den Standardwert für diesen Parameter bei, solange Sie keine Leistungsprobleme feststellen.

hadr_db_role - Rolle der HADR-Datenbank

Dieser Parameter gibt die aktuelle Rolle einer Datenbank an, unabhängig davon, ob sie online oder offline ist.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients

Parametertyp
Informativ

Gültige Werte: `STANDARD`, `PRIMARY` oder `STANDBY`.

Anmerkung: Wenn eine Datenbank aktiv ist, kann die HADR-Rolle der Datenbank auch mit dem Befehl **`GET SNAPSHOT FOR DATABASE`** ermittelt werden.

hadr_local_host - Name des lokalen Hosts für HADR

Mit diesem Parameter wird der lokale Host für die HADR-TCP-Kommunikation (High Availability Disaster Recovery) angegeben.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients

Parametertyp

- Konfigurierbar[6]

Standardwert
Null

6. Änderungen an diesem Parameter werden bei Datenbankaktivierung wirksam. Wenn die Datenbank bereits online ist, können Änderungen durch Stoppen und erneutes Starten von High Availability Disaster Recovery auf der primären Datenbank wirksam gemacht werden.

Es kann entweder ein Hostname oder eine IP-Adresse verwendet werden. Wenn ein Hostname angegeben wird und mehreren IP-Adressen zugeordnet ist, wird ein Fehler zurückgegeben und HADR wird nicht gestartet. Wenn der Hostname mehreren IP-Adressen zugeordnet ist (selbst wenn Sie den gleichen Hostnamen auf dem Primärsystem und dem Bereitschaftssystem angeben), besteht die Möglichkeit, dass das Primärsystem und das Bereitschaftssystem diesen Hostnamen verschiedenen IP-Adressen zuordnen, weil einige DNS-Server IP-Adresslisten in nicht deterministischer Reihenfolge zurückgeben.

Ein Hostname besitzt das Format: meinserver.ibm.com. Eine IP-Adresse besitzt das Format: "12.34.56.78".

Hinweise zurVerwendung

- Wenn das Primärsystem und das Bereitschaftssystem zwar starten, jedoch Verbindung zueinander herstellen und im DB2-Diagnoseprotokoll kein Fehleranzeiger erscheint, kann dies auf subtile Probleme bei der Auflösung der Hostnamen zurückzuführen sein. Wenn Sie HADR mithilfe von Hostnamen konfiguriert haben, verwenden Sie stattdessen die IP-Adressen in der Konfiguration.
- Wenn sich das Primärsystem bzw. das Bereitschaftssystem in einem privaten Netz hinter einer NAT-Einheit befindet (NAT = Network Address Translation), müssen Sie möglicherweise die Registrierdatenbankvariable **DB2_HADR_NO_IP_CHECK** auf ON setzen. Informationen hierzu finden Sie unter Unterstützung für HADR und NAT (Network Address Translation)

hadr_local_svc - Lokaler HADR-Servicename

Mit diesem Parameter wird der TCP-Servicename oder die Portnummer angegeben, für die der HADR-Prozess (High Availability Disaster Recovery) Verbindungen akzeptiert.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients

Parametertyp

- Konfigurierbar[7]

Standardwert
Null

Der Wert für **hadr_local_svc** auf dem primären Datenbanksystem oder dem Bereitschaftsdatenbanksystem darf nicht mit dem Wert für **svcename** bzw. **svcename** +1 auf den entsprechenden Hosts übereinstimmen.

Wenn Sie SSL verwenden, setzen Sie **hadr_local_svc** weder auf dem primären Datenbanksystem noch auf dem Bereitschaftsdatenbanksystem auf denselben Wert, der für **ssl_svcename** angegeben ist.

7. Änderungen an diesem Parameter werden bei Datenbankaktivierung wirksam. Wenn die Datenbank bereits online ist, können Änderungen durch Stoppen und erneutes Starten von High Availability Disaster Recovery auf der primären Datenbank wirksam gemacht werden.

hadr_peer_window - HADR-Peerfenster (Konfigurationsparameter)

Wenn Sie den Konfigurationsparameter **hadr_peer_window** auf einen Zeitwert ungleich null setzen, verhält sich das HADR-Datenbankpaar aus Primär- und Bereitschaftsdatenbank über den konfigurierten Zeitraum weiterhin so, als befände es sich im Peerzustand, wenn die Primärdatenbank die Verbindung zur Bereitschaftsdatenbank verliert. Dies hilft bei der Gewährleistung der Datenkonsistenz.

Konfigurationstyp
Datenbank

Parametertyp

- Konfigurierbar[8]

Standardwert [Bereich]
0 [0 – 4 294 967 295]

Maßeinheit
Sekunden

Hinweise:

- Der Wert muss bei Primär- und Bereitschaftsdatenbank übereinstimmen, sofern nicht der HADR-Modus für mehrere Bereitschaftsdatenbanken verwendet wird. Bei mehreren Bereitschaftsdatenbanken verwendet die Hauptbereitschaftsdatenbank die Einstellung der Primärdatenbank für **hadr_peer_window** und alle Einstellungen der Nebenbereitschaftsdatenbanken für **hadr_peer_window** werden ignoriert.
- Ein empfohlener Mindestwert beträgt 120 Sekunden.
- Wenn der Wert für **hadr_syncmode** auf ASYNC oder SUPERASYNC gesetzt ist, wird der Wert für **hadr_peer_window** ignoriert.
- Um eine Beeinträchtigung der Verfügbarkeit der Primärdatenbank zu vermeiden, wenn die Bereitschaftsdatenbank mit Absicht, zum Beispiel zu Wartungszwecken, heruntergefahren wird, wird das Peerfenster nicht aufgerufen, wenn die Bereitschaftsdatenbank explizit inaktiviert wird, während sich das HADR-Datenbankpaar im Peerzustand befindet.
- Der Befehl **TAKEOVER HADR** mit der Option **PEER WINDOW ONLY** startet eine Übernahmeoperation nur, wenn sich die HADR-Bereitschaftsdatenbank innerhalb des definierten Peerfensters befindet.
- Bei der Übernahmeoperation mit dem Parameter **hadr_peer_window** kann es zu einer fehlerhaften Funktionsweise kommen, wenn die Uhr der Primärdatenbank und die Uhr der Bereitschaftsdatenbank nicht mit einer Differenz von höchstens 5 Sekunden synchronisiert sind. Das heißt, die Operation ist möglicherweise erfolgreich, obwohl sie fehlschlagen sollte, oder sie schlägt fehl, obwohl sie erfolgreich sein sollte. Verwenden Sie einen Zeitsynchronisationsservice (z. B. NTP), um die Uhren über dieselbe Quelle zu synchronisieren.
- In der Bereitschaftsdatenbank basiert die Endzeit des Peerfensters auf der letzten Überwachungssignalnachricht (Heartbeat), die die Bereitschaftsdatenbank von der Primärdatenbank empfangen hat, und nicht auf dem Zeitpunkt, an dem die Bereitschaftsdatenbank die Verbindungstrennung feststellt.

8. Änderungen an diesem Parameter werden bei Datenbankaktivierung wirksam. Wenn die Datenbank bereits online ist, können Änderungen durch Stoppen und erneutes Starten von High Availability Disaster Recovery auf der primären Datenbank wirksam gemacht werden.

hadr_remote_host - Name des fernen HADR-Hosts

Mit diesem Parameter wird der TCP/IP-Hostname oder die IP-Adresse des fernen HADR-Datenbankservers (High Availability Disaster Recovery) angegeben.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients

Parametertyp

- Konfigurierbar[9]

Standardwert
Null

Ähnlich wie der Parameter **'hadr_local_host'** darf dieser Parameter nur einer IP-Adresse zugeordnet werden.

hadr_remote_inst - HADR-Instanzname des fernen Servers

Mit diesem Parameter wird der Instanzname des fernen Servers angegeben. Ebenso prüft HADR (High Availability Disaster Recovery), ob eine ferne Datenbank, die eine Verbindung anfordert, zu der deklarierten fernen Instanz gehört.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients

Parametertyp

- Konfigurierbar[10]

Standardwert
Null

hadr_remote_svc - Ferner HADR-Servicename

Mit diesem Parameter wird der TCP-Servicename oder die Portnummer angegeben, die vom fernen HADR-Datenbankserver (High Availability Disaster Recovery) verwendet wird.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients

Parametertyp

9. Änderungen an diesem Parameter werden bei Datenbankaktivierung wirksam. Wenn die Datenbank bereits online ist, können Änderungen durch Stoppen und erneutes Starten von High Availability Disaster Recovery auf der primären Datenbank wirksam gemacht werden.
10. Änderungen an diesem Parameter werden bei Datenbankaktivierung wirksam. Wenn die Datenbank bereits online ist, können Änderungen durch Stoppen und erneutes Starten von High Availability Disaster Recovery auf der primären Datenbank wirksam gemacht werden.

- Konfigurierbar[11]

Standardwert
Null

hadr_replay_delay - HADR-Wiedergabeverzögerung (Konfigurationsparameter)

Dieser Parameter gibt den Zeitraum in Sekunden an, der zwischen dem Zeitpunkt, an dem eine Transaktion auf der Primärdatenbank festgeschrieben wurde, und dem Zeitpunkt, an dem die Transaktion auf der Bereitschaftsdatenbank festgeschrieben wird, liegen muss.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
0 [0 to 2147483647]

Maßeinheit
Sekunden

Der Konfigurationsparameter **hadr_replay_delay** ermöglicht die *verzögerte Wiedergabe* in der HADR-Bereitschaftsdatenbank, d. h. die Protokolle der Primärdatenbank werden in der HADR-Bereitschaftsdatenbank absichtlich mit einer definierten Verzögerung wiedergegeben. Die Bereitschaftsdatenbank muss sich im Modus SUPERASYNC befinden, bevor die Verzögerung **hadr_replay_delay** mit einem Wert ungleich null definiert werden kann. Der Parameter kann in der Primärdatenbank nicht angegeben werden. Die verzögerte Wiedergabe muss für die Bereitschaftsdatenbank inaktiviert sein, damit sie die Funktion der Primärdatenbank übernehmen kann.

Wenn Sie die verzögerte Wiedergabe aktivieren, sollten Sie auch den Protokollspoolbetrieb aktivieren, indem Sie den Datenbankkonfigurationsparameter **hadr_spool_limit** definieren. Durch die beabsichtigte Verzögerung wird der Abstand zwischen der Protokollwiedergabeposition der Bereitschaftsdatenbank und der Protokollempfangsposition der Bereitschaftsdatenbank größer. Ohne Spoolbetrieb kann der Protokollempfang der Wiedergabe nur in einem Umfang vorausgreifen, der dem Umfang des Empfangspuffers entspricht. Mit Spoolbetrieb kann die Bereitschaftsdatenbank wesentlich mehr Protokolle über die Protokollwiedergabeposition hinaus empfangen und bietet somit bei einem Ausfall der Primärdatenbank einen höheren Schutz gegen Datenverlust. Dabei ist zu beachten, dass die Primärdatenbank in beiden Fällen aufgrund des obligatorischen Modus SUPERASYNC nicht durch die verzögerte Wiedergabe blockiert wird.

Wenn es auf der Primärdatenbank eine fehlerhafte Transaktion gegeben hat und dieser Fehler vor seiner Wiedergabe in der Bereitschaftsdatenbank bemerkt wird,

11. Änderungen an diesem Parameter werden bei Datenbankaktivierung wirksam. Wenn die Datenbank bereits online ist, können Änderungen durch Stoppen und erneutes Starten von High Availability Disaster Recovery auf der primären Datenbank wirksam gemacht werden.

können Sie die Datenbank mithilfe der verzögerten Wiedergabe bezogen auf einen Zeitpunkt aktualisierend wiederherstellen, der unmittelbar vor dem Festschreiben der fehlerhaften Transaktion liegt, und anschließend die aktualisierende Wiederherstellung stoppen, um die Daten, die auf der Primärdatenbank verloren gegangen sind, abzurufen.

Anmerkung: Eine Bereitschaftsdatenbank, bei der Sie die aktualisierende Wiedergabe aktiviert haben, kann im Bedarfsfall erst als neue Primärdatenbank eingesetzt werden, wenn Sie die verzögerte Wiedergabe inaktiviert haben. Dazu müssen Sie den Parameter **hadr_replay_delay** für die jeweilige Bereitschaftsdatenbank mit 0 definieren.

hadr_spool_limit – Begrenzung für HADR-Protokollspooling (Konfigurationsparameter)

Dieser Parameter ermittelt die maximale Protokolldatenmenge, die durch die HADR-Bereitschaftsfunktion auf die Platte übertragen wird.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients

Parametertyp
Konfigurierbar[12]

Standardwert [Bereich]
0 [-1 – 2.147.483.647]

Maßeinheit
Seiten (4 KB)

Der Konfigurationsparameter **hadr_spool_limit** ermöglicht den Spoolbetrieb für das Protokoll in der HADR-Bereitschaftsdatenbank. Der Protokoll-Spoolbetrieb ermöglicht das Fortsetzen von Transaktionen in der HADR-Primärdatenbank, ohne die Protokollwiedergabe der HADR-Bereitschaftsfunktion abzuwarten. Die von der Primärdatenbank gesendeten Protokolldaten werden im *Spoolbetrieb* auf die Platte der Bereitschaftsinstanz geschrieben, wenn eine Verzögerung bei der Protokollwiedergabe auftritt. Die Bereitschaftsinstanz kann die Protokolldaten später von der Platte lesen. Dadurch kann das System Lastspitzen im Transaktionsvolumen der Primärdatenbank besser ausgleichen bzw. Verzögerungen bei der Protokollwiedergabe in der Bereitschaftsdatenbank (z. B. verursacht durch die Wiedergabe bestimmter Protokollsatztypen).

Wenn Sie das Protokollspooling verwenden, müssen Sie sicherstellen, dass für den aktiven Protokollpfad der Bereitschaftsdatenbank adäquater Plattenspeicherplatz zur Verfügung steht. Es muss ausreichend Plattenspeicherplatz für die aktiven Protokolle zur Verfügung stehen. Dies wird durch die Konfigurationsparameter **logprimary**, **logsecond**, **logfilsiz** und **hadr_spool_limit** festgelegt.

Der Standardwert 0 bedeutet 'ohne Spoolbetrieb'. Dabei kann die Verzögerung der Protokollwiedergabe zwischen Primärdatenbank und Bereitschaftsdatenbank maxi-

12. Änderungen an diesem Parameter werden bei Datenbankaktivierung wirksam. Wenn die Datenbank bereits online ist, können Änderungen durch Stoppen und erneutes Starten von High Availability Disaster Recovery auf der primären Datenbank wirksam gemacht werden.

mal bis zur Größe des Protokollempfangspuffers zunehmen. Wenn der Puffer voll ist, werden neue Transaktionen in der Primärdatenbank möglicherweise blockiert, weil die Primärdatenbank keine weiteren Protokolldaten zum Bereitschaftssystem senden kann.

Der Wert -1 steht für unbegrenzten Spoolbetrieb (soweit der verfügbare Festplattenspeicher dies zulässt). Bei Verwendung eines hohen Werts für **hadr_spool_limit** sollten Sie bedenken, dass bei einem großen Abstand zwischen der Protokollwiedergabeposition des Primärsystems und der Protokollwiedergabeposition des Bereitschaftssystems die Funktionsübernahme deutlich länger dauern kann, weil das Bereitschaftssystem die Rolle als neue Bereitschaftsinstanz erst nach Beendigung der Wiedergabe der Spooling-Protokolle übernehmen kann.

Hinweis: Die Verwendung des Protokollspoolbetriebs stellt keine Gefährdung der von der HADR-Komponente bereitgestellten HADR-Schutzfunktion dar. Die Daten vom Primärsystem werden weiterhin unter Verwendung des angegebenen Synchronisationsmodus in Protokollform auf dem Bereitschaftssystem repliziert. Durch die Protokollwiedergabe tritt nur beim Anwenden der Daten auf die Tabellenbereiche eine Verzögerung ein.

hadr_syncmode - HADR-Synchronisationsmodus für Protokollierung im Peerstatus

Mit diesem Parameter wird der Synchronisationsmodus angegeben, der bestimmt, wie die Protokollschreiboperationen der Primärdatenbank mit der Bereitschaftsdatenbank synchronisiert werden, wenn sich die Systeme im Peerstatus befinden.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients

Parametertyp

- Konfigurierbar[13]

Standardwert [Bereich]
NEARSYNC [ASYNC, SUPERASYNC, SYNC]

Gültige Werte für diesen Parameter sind:

SYNC

Dieser Modus bietet den größten Schutz gegen Transaktionsverlust, jedoch auf Kosten einer längeren Transaktionsantwortzeit.

In diesem Modus wird das Schreiben von Protokollen nur dann als erfolgreich betrachtet, wenn die Protokolle in Protokolldateien in der Primärdatenbank geschrieben wurden und wenn die Primärdatenbank von der Bereitschaftsdatenbank eine Empfangsbestätigung erhalten hat, dass die Protokolle auch in Protokolldateien in der Bereitschaftsdatenbank geschrieben wurden. So wird sichergestellt, dass die Protokolldaten an beiden Standorten gespeichert werden.

13. Änderungen an diesem Parameter werden bei Datenbankaktivierung wirksam. Wenn die Datenbank bereits online ist, können Änderungen durch Stoppen und erneutes Starten von High Availability Disaster Recovery auf der primären Datenbank wirksam gemacht werden.

NEARSYNC

Dieser Modus bietet einen etwas geringeren Schutz gegen Transaktionsverlust, jedoch im Gegenzug auch eine kürzere Transaktionsantwortzeit als der Modus SYNC.

In diesem Modus wird das Schreiben von Protokollen nur dann als erfolgreich betrachtet, wenn die Protokolle in Protokolldateien in der Primärdatenbank geschrieben wurden und wenn die Primärdatenbank von der Bereitschaftsdatenbank eine Empfangsbestätigung erhalten hat, dass die Protokolle auch in den Hauptspeicher des Bereitschaftssystems geschrieben wurden. Zu Datenverlust kann es nur kommen, wenn beide Standorte gleichzeitig ausfallen und wenn der Bereitschaftsstandort nicht alle empfangenen Protokolldaten an nicht flüchtigen Speicher übertragen hat.

ASYNC

Im Vergleich zu den Modi SYNC und NEARSYNC hat der Modus ASYNC zwar kürzere Transaktionsantwortzeiten zur Folge, kann aber größere Transaktionsverluste zur Folge haben, wenn die Primärdatenbank ausfällt.

In diesem Modus wird das Schreiben von Protokollen nur dann als erfolgreich betrachtet, wenn die Protokolle in Protokolldateien in der Primärdatenbank geschrieben und an die TCP-Schicht der Hostmaschine des primären Systems übergeben wurden. Da das primäre System nicht auf eine Empfangsbestätigung vom Bereitschaftssystem wartet, können Transaktionen bereits als festgeschrieben betrachtet werden, während sie noch an das Bereitschaftssystem weitergeleitet werden.

SUPERASYNC

Dieser Modus hat die kürzeste Transaktionsantwortzeit; bei diesem Modus sind aber auch Transaktionsverluste am Wahrscheinlichsten, wenn das primäre System ausfällt. Dieser Modus ist nützlich, wenn Sie möchten, dass Transaktionen niemals geblockt werden; es kommt auch niemals zu ausgedehnten Antwortzeiten aufgrund von Netzunterbrechungen oder Überlastung.

In diesem Modus kann das HADR-Paar niemals den Status 'Peer' oder 'Unterbrochener Peer' aufweisen. Das Schreiben von Protokollen wird nur als erfolgreich betrachtet, wenn die Protokollsätze in Protokolldateien in der Primärdatenbank geschrieben wurden. Da das primäre System nicht auf eine Empfangsbestätigung vom Bereitschaftssystem wartet, können Transaktionen bereits als festgeschrieben betrachtet werden, während sie noch an das Bereitschaftssystem weitergeleitet werden.

In Abb. 52 auf Seite 888 sehen Sie, wann die Protokolle für Transaktionen auf der Basis des ausgewählten Synchronisationsmodus als erfolgreich betrachtet werden:

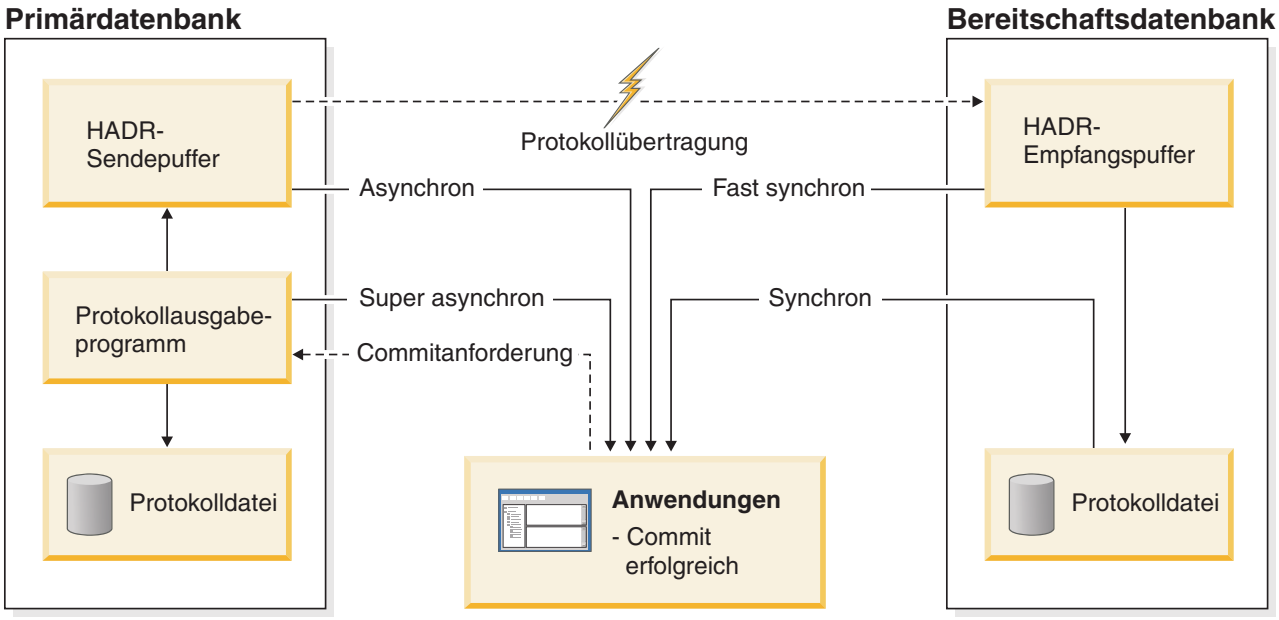


Abbildung 52. Synchronisationsmodi für HADR (High Availability and Disaster Recovery)

hadr_target_list - Liste der HADR-Ziele (Datenbankkonfigurationsparameter)

Dieser Parameter ermöglicht eine HADR-Ausführung im Modus für mehrere Bereitschaftsdatenbanken und gibt eine Liste mit bis zu drei *host*:*port*-Zielpaaren an, die als HADR-Bereitschaftsdatenbanken fungieren.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients

Parametertyp
Online konfigurierbar

Standardwert
NULL

Damit HADR im Modus für mehrere Bereitschaftsdatenbanken ausgeführt werden kann, muss der Konfigurationsparameter **hadr_target_list** bei allen HADR-Datenbanken definiert sein. Sie können mithilfe dieses Parameters nur vom Modus für eine Bereitschaftsdatenbank in den Modus für mehrere Bereitschaftsdatenbanken wechseln, wenn Sie zunächst die Primärdatenbank herunterfahren. Die *host*:*port*-Paare, die Sie für den Konfigurationsparameter **hadr_target_list** für eine Primärdatenbank angeben, bestimmen die Hosts, die als Bereitschaftsdatenbank für die betreffende Primärdatenbank eingesetzt werden. Die *host*:*port*-Paare in der Zielliste einer Bereitschaftsdatenbank geben die Hosts an, die als Bereitschaftsdatenbanken verwendet werden sollen, wenn die jeweilige Bereitschaftsdatenbank als neue HADR-Primärdatenbank eingesetzt wird.

Die Standardeinstellung NULL gibt an, dass die HADR-Ausführung im Modus für eine Bereitschaftsdatenbank erfolgt.

Wie bei den Datenbankkonfigurationsparametern **hadr_remote_host** und **hadr_local_host** können Sie auch für **hadr_target_list** einen Host über einen Hostnamen oder eine IP-Adresse angeben. Ein Hostname darf nur alphanumerische Zeichen, Gedankenstriche und Unterstreichungszeichen enthalten. Wie bei den Datenbankkonfigurationsparametern **hadr_remote_svc** und **hadr_local_svc** können Sie auch für den Konfigurationsparameter **hadr_target_list** einen Port über eine Portnummer oder einen Servicenamen angeben. Ein Servicename kann beliebige Zeichen enthalten. Hostnamen werden einer IP-Adresse zugeordnet und Servicenamen einer Portnummer. Die Werte, die Sie für den Konfigurationsparameter **hadr_target_list** angeben, können eine beliebige Kombination aus Hostnamen, Host-IP-Adressen, Servicenamen und Portwerten enthalten. Geben Sie die *host*:*port*-Paare im folgenden Format an:

host1:*port1*|*host2*:*port2*|*host3*:*port3*

Numerische IPv6-Adressen (IPv6 = Internet Protocol version 6) müssen Sie wie im folgenden Beispiel ersichtlich in eckige Klammern ([]) setzen, um den IP-Teil vom Portteil abzugrenzen:

```
[FEDC:BA98:7654:3210:FEDC:BA98:7654:3210]:4000
```

Der erste Eintrag in der Zielliste für eine HADR-Datenbank ist die *HADR-Hauptbereitschaftsdatenbank*, alle weiteren Einträge sind *HADR-Nebenbereitschaftsdatenbanken*. Sie können die Hauptbereitschaftsdatenbank und die Primärdatenbank für einen beliebigen Synchronisationsmodus konfigurieren. Der Synchronisationsmodus der Hauptbereitschaftsdatenbank wird jedoch durch die Einstellung für den Konfigurationsparameter **hadr_syncmode** der Primärdatenbank bestimmt. Der Synchronisationsmodus der Nebenbereitschaftsdatenbank ist immer SUPERASYNC. Wenn Sie HADR in der Primärdatenbank starten, werden die Werte für die Konfigurationsparameter **hadr_remote_host** und **hadr_remote_svc** automatisch wie die Parameter für die Hauptbereitschaftsdatenbank definiert, sofern die Registrierdatenbankvariable **DB2_HADR_NO_IP_CHECK** nicht von Ihnen mit ON definiert wurde.

Eine Symmetrie bzw. reziproke Einstellungen sind nicht erforderlich. Für Datenbank A kann Datenbank B als Hauptbereitschaftsdatenbank angegeben werden und für Datenbank B Datenbank C. Ein Rollentausch innerhalb der einzelnen Datenbankpaare muss jedoch möglich sein. Ist Datenbank B z. B. in der Zielliste von Datenbank A enthalten, muss Datenbank A in der Zielliste von Datenbank B aufgeführt sein.

Der Parameter **hadr_target_list** kann mit sofortiger Wirkung aktualisiert werden, während die Datenbank online ist. Wenn HADR aktiv ist, sind jedoch einige Einschränkungen zu beachten:

- Sie können die Hauptbereitschaftsdatenbank für die Primärdatenbank nicht ändern, ohne zunächst HADR auf der Primärdatenbank zu stoppen.
- Sie können eine Bereitschaftsdatenbank nicht aus der Liste entfernen, wenn diese Datenbank mit der Primärdatenbank verbunden ist. Trennen Sie die Verbindung zu einer Bereitschaftsdatenbank, indem Sie die betreffende Datenbank einfach deaktivieren. Anschließend können Sie die Datenbank in der Zielliste der Primärdatenbank entfernen.
- Der Konfigurationsparameter **hadr_target_list** für eine Bereitschaftsdatenbank kann nur dynamisch aktualisiert werden, wenn Sie die Funktion für Leseoperationen in der Bereitschaftsdatenbank aktiviert haben.

- Sie können die Primärdatenbank nicht aus der Zielliste einer Bereitschaftsdatenbank entfernen, wenn die Bereitschaftsdatenbank mit der Primärdatenbank verbunden ist.
- Bei den IP-Adressen in der Zielliste muss es sich entweder um IPv4-Adressen oder um IPv6-Adressen handeln. Eine Kombination dieser Adressen ist nicht möglich.

hadr_timeout - HADR-Zeitlimitwert

Dieser Parameter gibt die Zeit (in Sekunden) an, die der HADR-Prozess (High Availability Disaster Recovery) wartet, bevor ein Kommunikationsversuch als fehlgeschlagen eingestuft wird.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients

Parametertyp

- Konfigurierbar[14]

Standardwert [Bereich]
120 [1 - 4 294 967 295]

indexrec - Zeitpunkt für Indexneuerstellung

Dieser Parameter gibt an, wann der Datenbankmanager versucht, ungültige Indizes neu zu erstellen und ob eine Indexerstellung während einer aktualisierenden Recovery oder einer Wiedergabe des HADR-Protokolls in der Bereitschaftsdatenbank wiederholt wird.

Konfigurationstyp
Datenbank und Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]

UNIX Datenbankmanager
`restart` [`restart`; `restart_no_redo`; `access`; `access_no_redo`]

Windows Datenbankmanager
`restart` [`restart`; `restart_no_redo`; `access`; `access_no_redo`]

14. Änderungen an diesem Parameter werden bei Datenbankaktivierung wirksam. Wenn die Datenbank bereits online ist, können Änderungen durch Stoppen und erneutes Starten von High Availability Disaster Recovery auf der primären Datenbank wirksam gemacht werden.

Datenbank
Systemeinstellung verwenden [system; restart; restart_no_redo; access; access_no_redo]

Für diesen Parameter sind fünf Einstellungen möglich:

SYSTEM
Verwenden Sie die Systemeinstellung, die in der Konfigurationsdatei des Datenbankmanagers angegeben wurde, um festzulegen, wann ungültige Indizes neu erstellt werden und ob Protokollsätze zur Indexerstellung während einer aktualisierenden Recovery oder während einer Wiedergabe des HADR-Protokolls wiederholt werden müssen.

Anmerkung: Diese Einstellung ist nur für Datenbankkonfigurationen gültig.

ACCESS
Ungültige Indizes werden erneut erstellt, wenn zum ersten Mal auf die zugrunde liegende Tabelle zugegriffen wird. Alle vollständig protokollierten Indexerstellungen werden während einer aktualisierenden Recovery oder während einer Wiedergabe des HADR-Protokolls wiederholt. Wenn HADR gestartet wird und eine HADR-Übernahme stattfindet, werden alle ungültigen Indizes nach der Übernahme, wenn zum ersten Mal auf die zugrunde liegende Tabelle zugegriffen wird, neu erstellt.

ACCESS_NO_REDO
Ungültige Indizes werden erneut erstellt, wenn zum ersten Mal auf die zugrunde liegende Tabelle zugegriffen wird. Keine der vollständig protokollierten Indexerstellungen wird während einer aktualisierenden Recovery oder während einer Wiedergabe des HADR-Protokolls wiederholt; diese Indizes bleiben ungültig. Wenn HADR gestartet wird und eine HADR-Übernahme stattfindet, werden alle ungültigen Indizes nach der Übernahme, wenn zum ersten Mal auf die zugrunde liegende Tabelle zugegriffen wird, neu erstellt. Der Zugriff auf die zugrunde liegenden Tabellen in der neuen Primärdatenbank bewirkt einen Indexrebuild. Dies führt dazu, dass Protokollsätze geschrieben und an die neue Bereitschaftsdatenbank gesendet werden. Dies wiederum führt dazu, dass die Indizes in der Bereitschaftsdatenbank ungültig gemacht werden.

RESTART
Die Standardeinstellung für **indexrec**. Ungültige Indizes werden neu erstellt, wenn der Befehl **RESTART DATABASE** explizit oder implizit abgesetzt wird. Alle vollständig protokollierten Indexerstellungen werden während einer aktualisierenden Recovery oder während einer Wiedergabe des HADR-Protokolls wiederholt. Wenn HADR gestartet wird und eine HADR-Übernahme stattfindet, werden alle ungültigen Indizes nach der Übernahme neu erstellt.

Anmerkung: In einer DB2 pureScale-Umgebung werden Indizes nur bei einer Recovery nach Absturz einer Gruppe erneut erstellt, nicht im Rahmen der Recovery nach Absturz eines Members.

Anmerkung: Wenn beim abnormalen Beenden einer Datenbank, die mit Anwendungen verbunden ist, der Parameter autorestart aktiviert wird, wird beim Herstellen einer Verbindung von einer Anwendung zu einer Datenbank implizit der Befehl **RESTART DATABASE** abgesetzt. Wenn der Befehl nicht abgesetzt wird, werden die ungültigen Indizes erneut erstellt, sobald der nächste Zugriff auf die zugrunde liegende Tabelle erfolgt.

RESTART_NO_REDO

Ungültige Indizes werden neu erstellt, wenn der Befehl **RESTART DATABASE** explizit oder implizit abgesetzt wird. Keine der vollständig protokollierten Indexerstellungen wird während einer aktualisierenden Recovery oder während einer Wiedergabe des HADR-Protokolls wiederholt; stattdessen werden diese Indizes neu erstellt, wenn die aktualisierende Recovery beendet wird oder wenn die HADR-Übernahme stattfindet. Die Übernahme bewirkt einen Indexrebuild für die zugrunde liegenden Tabellen in der Primärdatenbank. Dies führt dazu, dass Protokollsätze geschrieben und an die neue Bereitschaftsdatenbank gesendet werden. Dies wiederum führt dazu, dass die Indizes in der Bereitschaftsdatenbank ungültig gemacht werden.

Wenn beim abnormalen Beenden einer Datenbank, die mit Anwendungen verbunden ist, der Parameter `autorestart` aktiviert wird, wird beim Herstellen einer Verbindung von einer Anwendung zu einer Datenbank implizit der Befehl **RESTART DATABASE** abgesetzt. Wenn der Befehl nicht abgesetzt wird, werden die ungültigen Indizes erneut erstellt, sobald der nächste Zugriff auf die zugrunde liegende Tabelle erfolgt.

Indizes können ungültig werden, wenn nicht behebbare Plattenfehler auftreten. Wenn dabei die Daten selbst beschädigt werden, können die Daten verloren gehen. Wird jedoch ein Index beschädigt, kann der Index durch Neuerstellung wiederhergestellt werden. Wird ein Index neu erstellt, während Benutzer mit der Datenbank verbunden sind, können zwei Probleme auftreten:

- Während der Erstellung der Indexdatei könnte es zu einer unerwarteten Verschlechterung der Antwortzeit kommen. Benutzer, die auf die Tabelle zugreifen und diesen bestimmten Index verwenden, müssen auf die Neuerstellung des Index warten.
- Nach der Indexneuerstellung könnten unerwartete, aktive Sperren beibehalten werden, insbesondere wenn die Benutzertransaktion, die die Indexneuerstellung ausgelöst hat, keine COMMIT- oder ROLLBACK-Operation ausgeführt hat.

Empfehlung: Die beste Option für diesen Parameter auf einem Server mit vielen Benutzern, wenn die Zeit für den Neustart keine kritische Rolle spielt, ist die Indexneuerstellung beim Neustart der Datenbank (**DATABASE RESTART**) als Teil eines Vorgangs, mit dem die Datenbank nach einem Systemabsturz wiederhergestellt und online verfügbar gemacht wird.

Wenn dieser Parameter auf `ACCESS` oder auf `ACCESS_NO_REDO` gesetzt wird, verschlechtert sich während der Indexneuerstellung die Leistung des Datenbankmanagers. Jeder Benutzer, der auf den Index oder die Tabelle zugreift, der bzw. die gerade neu erstellt wird, muss zunächst auf die Beendigung der Indexneuerstellung warten.

Wenn dieser Parameter auf `RESTART` gesetzt ist, dauert der Neustart der Datenbank wegen der Indexneuerstellung länger, jedoch wird die normale Verarbeitung nicht mehr beeinträchtigt, sobald die Datenbank wieder online verfügbar ist.

Anmerkung: Bei der Datenbankrecovery werden alle ausführbaren Dateien von SQL-Prozeduren im Dateisystem entfernt, die zu der gerade wiederhergestellten Datenbank gehören. Wenn der Parameter **indexrec** auf `RESTART` gesetzt ist, werden alle ausführbaren Dateien von SQL-Prozeduren aus dem Datenbankkatalog extrahiert und in das Dateisystem zurückgestellt, wenn die nächste Verbindung zur Datenbank hergestellt wird. Wenn der Parameter **indexrec** nicht auf `RESTART` gesetzt ist, wird die ausführbare Datei einer SQL-Prozedur nur bei der ersten Ausführung der SQL-Prozedur in das Dateisystem extrahiert.

Der Unterschied zwischen den Werten RESTART und RESTART_NO_REDO bzw. zwischen den Werten ACCESS und ACCESS_NO_REDO ist nur von Bedeutung, wenn die vollständige Protokollierung für Indexerstellungsoperationen (z. B. **CREATE INDEX** und **REORG INDEX**) oder für eine Indexneuerstellung aktiviert ist. Sie können die Protokollierung aktivieren, indem Sie den Datenbankkonfigurationsparameter **logindexbuild** oder LOG INDEX BUILD beim Ändern einer Tabelle aktivieren. Wenn Sie **indexrec** entweder auf RESTART oder auf ACCESS setzen, kann für Operationen, die eine protokollierte Indexerstellung einbeziehen, eine aktualisierende Recovery durchgeführt werden, ohne dass das Indexobjekt einen ungültigen Status aufweist; dies hätte zur Folge, dass der Index später neu erstellt werden müsste.

jdk_64_path - Installationspfad für 64-Bit Software Developer's Kit für Java auf DAS

Dieser Parameter gibt das Verzeichnis an, in dem das 64-Bit-Software Developer's Kit (SDK) für Java installiert ist, das zu verwenden ist, um DB2-Verwaltungsserverfunktionen auszuführen.

Konfigurationstyp
DB2-Verwaltungsserver

Gilt für
DB2-Verwaltungsserver

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
Null [beliebiger gültiger Pfad]

Anmerkung: Dieser Parameter unterscheidet sich vom Konfigurationsparameter **jdk_path**, der ein 32-Bit-SDK für Java angibt.

Die vom Java-Interpreter verwendeten Umgebungsvariablen werden aus dem Wert dieses Parameters berechnet. Dieser Parameter wird nur auf den Plattformen verwendet, die 32- und 64-Bit-Instanzen unterstützen.

Bei den Plattformen handelt es sich um folgende:

- 64-Bit-Kernel für AIX, HP-UX und Solaris Operating System
- 64-Bit-Windows unter x64 und IPF
- 64-Bit-Linux-Kernel auf AMD64- und Intel EM64T-Systemen (x64), POWER und zSeries

Bei allen anderen Plattformen wird nur **jdk_path** verwendet.

Da es für diesen Parameter keinen Standardwert gibt, sollten Sie beim Installieren des SDK für Java einen Wert angeben.

Dieser Parameter kann nur über einen Befehlszeilenprozessor (CLP) der Version 8 aktualisiert werden.

locklist - Maximaler Speicher für Sperrenliste

Dieser Parameter gibt die Speichermenge an, die für die Sperrenliste zugeordnet wird. Für jede Datenbank gibt es eine Sperrenliste, die die Sperren aller gleichzeitig mit der Datenbank verbundenen Anwendungen enthält.

Konfigurationstyp
Datenbank

Parametertyp

- Online konfigurierbar
- Nach Member in einer DB2 pureScale-Umgebung konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
`Automatic` [4 - 134217728]

Anmerkung: Der Standardwert unterliegt nach der ursprünglichen Datenbankerstellung Änderungen durch den DB2 Configuration Advisor.

Maßeinheit
Seiten (4 KB)

Zuordnung
Wenn die erste Anwendung die Verbindung zur Datenbank herstellt

Freigabe
Wenn die letzte Anwendung die Verbindung zur Datenbank beendet

Das Sperren ist eine Funktion des Datenbankmanagers zur Steuerung des gleichzeitigen Zugriffs auf Daten der Datenbank durch mehrere Anwendungen. Sowohl Zeilen als auch Tabellen können gesperrt werden. Der Datenbankmanager fordert eventuell auch Sperren zur internen Verwendung an.

Der Wert `AUTOMATIC` für diesen Parameter aktiviert ihn für die automatische Leistungsoptimierung. Dadurch ist die Speicheroptimierungsfunktion in der Lage, die durch diesen Parameter gesteuerte Größe des Hauptspeicherbereichs an geänderte Auslastungsanforderungen je nach Bedarf dynamisch anzupassen. Da die Speicheroptimierungsfunktion Speicherressourcen auf verschiedene Speicherverbraucher verteilt, müssen mindestens zwei Speicherverbraucher für die automatische Optimierung aktiviert sein, damit eine automatische Optimierung erfolgen kann.

Obwohl der Wert des Parameters **`locklist`** zusammen mit dem Parameter **`maxlocks`** optimiert werden kann, bewirkt die Inaktivierung der automatischen Optimierung des Parameters **`locklist`** nicht automatisch die Inaktivierung der automatischen Optimierung des Parameters **`maxlocks`**. Die Aktivierung der automatischen Optimierung für den Parameter **`locklist`** führt automatisch zu einer Aktivierung der automatischen Optimierung des Parameters **`maxlocks`**.

Die automatische Optimierung dieses Konfigurationsparameters findet nur statt, wenn der Speicher mit automatischer Leistungsoptimierung für die Datenbank aktiviert ist (der Konfigurationsparameter **`self_tuning_mem`** muss auf `ON` gesetzt sein).

Auf allen Plattformen erfordert jede Sperre 128 oder 256 Byte der Sperrenliste, und zwar in Abhängigkeit davon, ob für das Objekt andere Sperren aktiv sind:

- 256 Byte sind erforderlich, um eine Sperre für ein Objekt zu aktivieren, für das keine anderen Sperren aktiv sind.
- 128 Byte sind erforderlich, um eine Sperre für ein Objekt einzutragen, für das bereits eine vorhandene Sperre aktiv ist.

Wenn der Prozentsatz der Sperrenliste, der von einer Anwendung verwendet wird, den Wert des Parameters **`maxlocks`** erreicht, führt der Datenbankmanager eine

Sperreneskalation von Zeilenebene auf Tabellenebene für die Sperren aus, die von dieser Anwendung aktiviert wurden. Diese Berechnung ist ein Näherungswert, wobei nur von gemeinsam genutzten Sperren ausgegangen wird. Der Prozentsatz der verwendeten Sperrenliste wird durch Multiplizieren der Anzahl von Sperren durch die Anwendung mit dem Wert, der für eine Sperre für ein Objekt erforderlich ist, für das andere Sperren existieren, berechnet. Obwohl der Eskalationsprozess an sich nicht viel Zeit in Anspruch nimmt, wird durch Sperren ganzer Tabellen (im Vergleich zum Sperren einzelner Zeilen) der gemeinsame Zugriff eingeschränkt und die allgemeine Datenbankleistung kann bei nachfolgenden Zugriffen auf die betroffenen Tabellen beeinträchtigt werden. Es wird empfohlen, die Größe der Sperrenliste folgendermaßen klein zu halten:

- Führen Sie häufig Commits aus, um Sperren freizugeben.
- Wenn viele Aktualisierungen ausgeführt werden sollen, sperren Sie vor den Aktualisierungen die ganze Tabelle (mit der SQL-Anweisung LOCK TABLE). Dadurch wird nur eine Sperre verwendet und der Zugriff anderer auf die zu aktualisierenden Daten verhindert. Der gemeinsame Zugriff auf die Daten wird jedoch eingeschränkt.

  Sie können die Option LOCKSIZE der Anweisung ALTER TABLE auch verwenden, um das Sperren einer bestimmten Tabelle zu steuern.

  Die Verwendung der Isolationsstufe RR (Wiederholtes Lesen) kann zu einer automatischen Tabellensperre führen.
- Verwenden Sie die Isolationsstufe der Cursorstabilität (CS), wenn möglich, um die Anzahl der Sperren für gemeinsamen Zugriff zu verringern. Wenn dadurch die durch die Anwendung festgelegten Integritätsanforderungen nicht beeinträchtigt werden, verwenden Sie anstatt der Cursorstabilität die Isolationsstufe UR (Nicht festgeschriebener Lesevorgang), um die Anzahl der Sperren weiter zu verringern.
- Setzen Sie **locklist** auf AUTOMATIC. Die Sperrenliste wird synchron vergrößert, um eine Sperreneskalation bzw. ein vollständiges Füllen der Sperrenliste zu vermeiden.

Wenn die Sperrenliste voll ist, kann die Leistung herabgesetzt werden, da die Sperreneskalation mehr Tabellensperren und weniger Zeilensperren erzeugt und auf diese Weise den gemeinsamen Zugriff auf gemeinsam benutzte Objekte in der Datenbank einschränkt. Zudem kann es mehr Deadlocks zwischen Anwendungen geben (da sie alle auf eine verringerte Anzahl von Tabellensperren warten), was dazu führt, dass Transaktionen mit ROLLBACK rückgängig gemacht werden. Ihre Anwendung empfängt einen SQLCODE-Wert -912, wenn die maximale Anzahl von Sperranforderungen für die Datenbank erreicht wurde.

Empfehlung: Wenn Sperreneskalationen zu Leistungseinbußen führen, müssen Sie eventuell den Wert dieses Parameters oder den Wert des Parameters **maxlocks** erhöhen. Mit dem Datenbanksystemmonitor können Sie feststellen, ob Sperreneskalationen auftreten. Weitere Informationen finden Sie in der Beschreibung des Monitorelements **lock_escals** (Sperreneskalationen).

Die folgenden Schritte können Ihnen bei der Ermittlung der Anzahl der Seiten, die für Ihre Sperrenliste erforderlich sind, vielleicht helfen:

1. Berechnen Sie eine Untergrenze für die Größe der Sperrenliste, und verwenden Sie dazu *eine* der folgenden Formeln, abhängig von Ihrer jeweiligen Umgebung:
   a.
   ```
   (512 * 128 * maxappls) / 4096
   ```
   b. Bei aktiviertem Konzentrator:

```
(512 * 128 * max_coordagents) / 4096
```

c. In einer partitionierten Datenbank mit aktiviertem Konzentrator:

```
(512 * 128 * max_coordagents * Anzahl Datenbankpartitionen) / 4096
```

Dabei ist 512 ein Schätzwert für die durchschnittliche Anzahl von Sperren pro Anwendung und 128 ist die Anzahl der benötigten Byte für jede Sperre eines Objekts, für das bereits eine Sperre aktiv ist.

2. Berechnen Sie eine Obergrenze für die Größe der Sperrenliste:

```
(512 * 256 * maxappls) / 4096
```

In dieser Formel ist 256 die Anzahl der benötigten Byte für die erste Sperre eines Objekts.

Anmerkung: Definieren Sie in einer DB2 pureScale-Umgebung den Konfigurationsparameter **locklist** so, dass der Wert dem Wert für die Obergrenze des Werts, der in diesem Schritt berechnet wird, und 3 % der Gesamtanzahl aller Seiten des Pufferpools entspricht, die in der zu diesem Zeitpunkt verbundenen Datenbank vorliegen.

3. Schätzen Sie das Aufkommen an gemeinsamem Zugriff durch Anwendungen auf Ihre Daten, und wählen Sie nach Ihren Schätzungen einen Anfangswert für **locklist**, der zwischen der berechneten Ober- und Untergrenze liegt.
4. Mit dem Datenbanksystemmonitor können Sie, wie im folgenden Absatz beschrieben, den Wert dieses Parameters optimieren.

Wenn **maxappls** oder **max_coordagents** in Ihrem gültigen Szenario auf AUTOMATIC gesetzt ist, sollten Sie auch **locklist** auf AUTOMATIC setzen.

Sie können mit dem Datenbanksystemmonitor die maximale Anzahl der von einer bestimmten Transaktion aktivierten Sperren feststellen. Weitere Informationen finden Sie in der Beschreibung des Monitorelements **locks_held_top** (maximale Anzahl gehaltener Sperren).

Anhand dieser Informationen können Sie die geschätzte Anzahl der Sperren pro Anwendung bestätigen oder anpassen. Um diese Auswertung durchzuführen, müssen Sie mehrere Probeanwendungen ausführen und dabei beachten, dass die Monitordaten auf einer Transaktionsebene und nicht auf einer Anwendungsebene geliefert werden.

Der Wert des Parameters **locklist** sollte eventuell auch dann heraufgesetzt werden, wenn der Parameter **maxappls** erhöht wird oder wenn die Anwendungen, die ausgeführt werden, nur relativ selten Commits ausführen.

Wenn Sie diesen Parameter geändert haben, sollten Sie für die Anwendungen eventuell einen Rebind durchführen (mit dem Befehl **REBIND**).

locktimeout - Zeitlimit für Sperren

Mit diesem Parameter wird die Anzahl der Sekunden angegeben, die eine Anwendung auf eine Sperre wartet; dadurch werden globale Deadlocks für Anwendungen vermieden.

Konfigurationstyp
Datenbank

Parametertyp

- Konfigurierbar

Standardwert [Bereich]
-1 [-1; 0 - 32 767]

Maßeinheit
Sekunden

Wenn dieser Parameter auf 0 gesetzt wird, wird nicht auf Sperren gewartet. In diesem Fall erhält die Anwendung, wenn zum Zeitpunkt der Anforderung keine Sperre verfügbar ist, sofort die Rückmeldung -911.

Wenn dieser Parameter auf den Wert -1 gesetzt wird, wird die Überwachung des Zeitlimits für Sperren inaktiviert. In diesem Fall wird auf eine Sperre gewartet (wenn zum Zeitpunkt der Anforderung keine verfügbar ist), bis eines der folgenden Ereignisse eintritt:

- Die Sperre wird erteilt
- Ein Deadlock tritt ein

Anmerkung: Der Wert, den Sie für diesen Konfigurationsparameter eingeben, wird nicht zum Steuern von Überschreitungen des Sperrzeitlimits für Zieltabellen für Ereignismonitore verwendet. Ereignismonitore verwenden eine separate, nicht konfigurierbare Einstellung, die bei Sperren für Ereignismonitortabellen zu Überschreitungen des Zeitlimits führt.

Empfehlung: In einer Umgebung, in der Transaktionen verarbeitet werden (OLTP-Umgebung), können Sie einen Anfangswert von 30 Sekunden verwenden. In einer Umgebung, in der nur Abfragen durchgeführt werden, können Sie mit einem höheren Wert beginnen. In beiden Fällen sollten Sie diesen Parameter jedoch mithilfe von Vergleichstests (Benchmark-Tests) optimieren.

Der Wert sollte so eingestellt werden, dass schnell erkannt wird, wenn Wartezeiten aufgrund außergewöhnlicher Situationen auftreten, wie z. B. eine blockierte Transaktion (weil ein Benutzer seine Workstation verlassen hat). Sie sollten den Wert so hoch festlegen, dass gültige Sperrenanforderungen das Zeitlimit nicht aufgrund von Spitzenbelastungen überschreiten, da bei hoher Systemauslastung länger auf Sperren gewartet werden muss.

Mit dem Datenbanksystemmonitor können Sie die Häufigkeit verfolgen, mit der eine Anwendung (eine Verbindung) das Zeitlimit für Sperren überschreitet, oder erkennen, dass eine Datenbank eine Zeitlimitüberschreitung für alle verbundenen Anwendungen festgestellt hat.

Hohe Werte für das Monitorelement **lock_timeout** (Anzahl Zeitlimitüberschreitungen für Sperren) können folgende Ursachen haben:

- Ein zu niedriger Wert für diesen Konfigurationsparameter
- Eine Anwendung (Transaktion), die Sperren über einen längeren Zeitraum aufrechterhält. Mithilfe des Datenbanksystemmonitors können Sie solche Anwendungen näher untersuchen.
- Ein Problem in Bezug auf den gleichzeitigen Zugriff, verursacht durch Sperreneskalationen (von Sperren auf Zeilenebene zu Sperren auf Tabellenebene).

log_appl_info - Protokollsatz für Anwendungsinformationen (Datenbankkonfigurationsparameter)

Dieser Parameter gibt an, dass der Protokollsatz für Anwendungsinformationen zu Beginn jeder Aktualisierungstransaktion geschrieben wird.

Konfigurationstyp
Datenbank

Parametertyp
Konfigurierbar

Standardwert [Bereich]
No [Yes; No]

Die Einstellung YES bedeutet, dass zu Beginn jeder Aktualisierungstransaktion ein weiterer Protokollsatz hinzugefügt wird. Der Protokollsatz mit Anwendungsinformationen wird für die Transaktion verwendet (siehe Kopfinformationen des Protokollsatzes).

Aktivieren Sie den Konfigurationsparameter (**log_appl_info**), wenn zu Replikationszwecken Tabellen mit der Option DATA CAPTURE CHANGES erstellt werden (oder zu anderen Zwecken, z. B. für die Prüffunktion).

log_ddl_stmts - DDL-Anweisungen protokollieren (Datenbankkonfigurationsparameter)

Dieser Parameter gibt an, dass zusätzliche Informationen zu DDL-Anweisungen (DDL = Data Definition Language) in das Protokoll geschrieben werden.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
No [Yes; No]

Die Standardeinstellung des Konfigurationsparameters **log_ddl_stmts** verhindert, dass ein Zusatzaufwand für das Schreiben von zusätzlichen Protokollsätzen für DDL-Operationen entsteht. Definieren Sie diesen Parameter mit YES, wenn DDL-Operationen repliziert werden müssen und ein Erfassungsprogramm für die Replikation verwendet wird, um Änderungen im Protokoll zu erfassen.

log_retain_status - Statusanzeiger für Beibehalten der Protokolle

Wenn der Datenbankkonfigurationsparameter **logarchmeth1** auf den Wert **logretain** gesetzt ist, wird im Parameter 'log_retain_status' der Wert RECOVERY angezeigt; andernfalls zeigt dieser Parameter den Wert NO an.

Konfigurationstyp
Datenbank

Parametertyp
Informativ

logarchcompr1 - Komprimierung für primäre archivierte Protokolldateien (Konfigurationsparameter)

Dieser Parameter gibt an, ob die Protokolldateien, die in das primäre Archivziel für Protokolle geschrieben werden, komprimiert werden.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Clients
- Datenbankserver mit lokalen Clients
- Partitionierte Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
OFF [ON]

OFF Gibt an, dass Protokolldateien, die in das primäre Archivziel für Protokolle geschrieben werden, nicht komprimiert werden.

ON Gibt an, dass Protokolldateien, die in das primäre Archivziel für Protokolle geschrieben werden, komprimiert werden. Bei dynamischer Festlegung werden bereits archivierte Protokolldateien nicht komprimiert.

Hinweis:

- Wenn Sie für den Konfigurationsparameter **logarchmeth1** einen anderen Wert als DISK, TSM oder VENDOR einstellen, hat die Komprimierung des Protokollarchivs unabhängig von der Einstellung für den Parameter **logarchcompr1** keine Auswirkungen.

logarchcompr2 - Komprimierung für sekundäre archivierte Protokolldateien (Konfigurationsparameter)

Dieser Parameter gibt an, ob die Protokolldateien, die in das sekundäre Archivziel für Protokolle geschrieben werden, komprimiert werden.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Clients
- Datenbankserver mit lokalen Clients
- Partitionierte Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
OFF [ON]

OFF Gibt an, dass Protokolldateien, die in das sekundäre Archivziel für Protokolle geschrieben werden, nicht komprimiert werden.

ON Gibt an, dass Protokolldateien, die in das sekundäre Archivziel für Protokolle geschrieben werden, komprimiert werden. Bei dynamischer Festlegung werden bereits archivierte Protokolldateien nicht komprimiert.

Hinweis:

- Wenn Sie für den Konfigurationsparameter **logarchmeth2** einen anderen Wert als DISK, TSM oder VENDOR einstellen, hat die Komprimierung des Protokollarchivs unabhängig von der Einstellung für den Parameter **logarchcompr2** keine Auswirkungen.

logarchmeth1 - Primäre Protokollarchivierungsmethode

Dieser Parameter gibt den Datenträgertyp des primären Ziels für Protokolle an, die aus dem aktuellen Protokollpfad archiviert werden.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Clients
- Datenbankserver mit lokalen Clients
- Partitionierte Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
OFF [LOGRETAIN, USEREXIT, DISK, TSM, VENDOR]

OFF Gibt an, dass die Protokollarchivierungsmethode nicht verwendet wird. Wenn Sie sowohl den Konfigurationsparameter **logarchmeth1** als auch **logarchmeth2** auf den Standardwert OFF setzen, bedeutet dies, dass die Datenbank Umlaufprotokolle verwendet. Sie kann in diesem Fall nicht aktualisierend wiederhergestellt werden.

LOGRETAIN
Gibt an, dass aktive Protokolldateien gespeichert und als Online-Archivprotokolldateien in einer aktualisierenden Recovery eingesetzt werden.

USEREXIT
Gibt an, dass die Protokollierung mit Protokollspeicherung ausgeführt werden soll und dass ein Benutzerexitprogramm für die Archivierung und das Abrufen von Protokolldateien verwendet werden soll. Protkolldateien werden archiviert, wenn sie voll sind. Sie werden abgerufen, wenn das Dienstprogramm **ROLLFORWARD** sie für den Restore einer Datenbank benötigt.

DISK Sie müssen diesem Wert einen Doppelpunkt (:) und anschließend einen vollständig qualifizierten und vorhandenen Namen eines Pfades nachstellen, in dem die Protokolldateien archiviert werden sollen. Wenn Sie beispielsweise den Konfigurationsparameter **logarchmeth1** auf DISK:/u/dbuser/archived_logs setzen, werden die Archivprotokolldateien in das Verzeichnis /u/dbuser/archived_logs/*instanz*/*datenbankname*/*knotenname*/*protokolldatenstrom*/*ketten-id*/ gestellt.

Anmerkung: Wenn Sie auf Band archivieren, können Sie zum Speichern und Abrufen der Protokolldateien das Dienstprogramm **db2tapemgr** verwenden.

TSM Wenn dieser Wert ohne weitere Konfigurationsparameter angegeben wird, gibt dies an, dass Protokolldateien unter Verwendung der Standardmanagementklasse auf dem lokalen TSM-Server archi-

viert werden. Wenn Sie dieser Option einen Doppelpunkt (:) und eine TSM-Managementklasse nachstellen, werden die Protokolldateien unter Verwendung der angegebenen Managementklasse archiviert.

Wenn Protokolle mit TSM archiviert werden, versucht TSM vor der Verwendung der durch den Datenbankkonfigurationsparameter angegebenen Managementklasse, das Objekt an die Managementklasse zu binden, die Sie in der Liste INCLUDE-EXCLUDE in der Datei mit den TSM-Clientoptionen angegeben haben. Wird keine Übereinstimmung gefunden, wird die TSM-Standardmanagementklasse verwendet, die Sie auf dem TSM-Server angegeben haben. Anschließend bindet TSM das Objekt erneut an die durch den Datenbankkonfigurationsparameter angegebene Managementklasse-.Daher muss sowohl die Standardmanagementklasse als auch die von Ihnen durch den Datenbankkonfigurationsparameter angegebene Managementklasse eine Archivierungskopiengruppe enthalten, da andernfalls die Archivierungsoperation fehlschlägt. Beispiele für TSM-Einträge:

- Bei Angabe einer Managementklasse: `db2 update db cfg for mydb using logarchmeth1 TSM:DB2_LOGS`
- Ohne Angabe einer Managementklasse: `db2 update db cfg for mydb using logarchmeth1 TSM`

VENDOR Gibt an, dass zur Archivierung der Protokolldateien eine Bibliothek eines anderen Anbieters verwendet wird. Sie müssen diesem Wert einen Doppelpunkt (:) und den Namen der Bibliothek nachstellen. Die APIs in der Bibliothek müssen die Backup- und Restore-APIs für Produkte anderer Anbieter verwenden. Beispiel für den Eintrag eines anderen Anbieters: `db2 update db cfg for mydb using logarchmeth1 VENDOR:/home/dbuser/vendorLib/<library name>`

Hinweise:

- Wenn Sie entweder den Konfigurationsparameter **logarchmeth1** oder **logarchmeth2** auf einen anderen Wert als `OFF` festlegen, ist die Datenbank für die aktualisierende Recovery konfiguriert.
- Wenn Sie die Option `userexit` oder `logretain` für den Konfigurationsparameter **logarchmeth1** verwenden, müssen Sie den Konfigurationsparameter **logarchmeth2** auf `OFF` setzen.
- Verwenden Sie die API **db2CfgSet**, um einen Archivpfad zuzuordnen, der ein Leerzeichen enthält.

logarchmeth2 - Sekundäre Protokollarchivierungsmethode

Dieser Parameter gibt den Datenträgertyp des sekundären Ziels für Protokolle an, die aus dem aktuellen Protokollpfad oder aus dem Pfad für Protokollspiegelung archiviert wurden.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Clients
- Datenbankserver mit lokalen Clients
- Partitionierte Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
OFF [DISK, TSM, VENDOR]

OFF Gibt an, dass die Protokollarchivierungsmethode nicht verwendet wird. Wenn Sie sowohl den Konfigurationsparameter **logarchmeth1** als auch **logarchmeth2** auf den Wert OFF setzen, bedeutet dies, dass die Datenbank Umlaufprotokolle verwendet. Sie kann in diesem Fall nicht aktualisierend wiederhergestellt werden. Dies ist der Standardwert.

DISK Sie müssen diesem Wert einen Doppelpunkt (:) und anschließend einen vollständig qualifizierten und vorhandenen Namen eines Pfades nachstellen, in dem die Protokolldateien archiviert werden sollen. Wenn Sie beispielsweise den Konfigurationsparameter **logarchmeth1** auf DISK:/u/dbuser/archived_logs setzen, werden die Archivprotokolldateien in das Verzeichnis /u/dbuser/archived_logs/*instanz*/*datenbankname*/*knotenname*/*ketten-id*/ gestellt.

Anmerkung: Wenn Sie auf Band archivieren, können Sie zum Speichern und Abrufen der Protokolldateien das Dienstprogramm **db2tapemgr** verwenden.

TSM Wenn dieser Wert ohne weitere Konfigurationsparameter angegeben wird, gibt dies an, dass Protokolldateien unter Verwendung der Standardmanagementklasse auf dem lokalen TSM-Server archiviert werden. Wenn Sie dieser Option einen Doppelpunkt (:) und eine TSM-Managementklasse nachstellen, werden die Protokolldateien unter Verwendung der angegebenen Managementklasse archiviert.

Wenn Protokolle mit TSM archiviert werden, versucht TSM vor der Verwendung der durch den Datenbankkonfigurationsparameter angegebenen Managementklasse, das Objekt an die Managementklasse zu binden, die Sie in der Liste INCLUDE-EXCLUDE in der Datei mit den TSM-Clientoptionen angegeben haben. Wird keine Übereinstimmung gefunden, wird die TSM-Standardmanagementklasse verwendet, die Sie auf dem TSM-Server angegeben haben. Anschließend bindet TSM das Objekt erneut an die durch den Datenbankkonfigurationsparameter angegebene Managementklasse-.Daher muss sowohl die Standardmanagementklasse als auch die von Ihnen durch den Datenbankkonfigurationsparameter angegebene Managementklasse eine Archivierungskopiengruppe enthalten, da andernfalls die Archivierungsoperation fehlschlägt.

VENDOR
Gibt an, dass zur Archivierung der Protokolldateien die Bibliothek eines anderen Anbieters verwendet wird. Sie müssen diesem Wert einen Doppelpunkt (:) und den Namen der Bibliothek nachstellen. Die APIs in der Bibliothek müssen die Backup- und Restore-APIs für Produkte anderer Anbieter verwenden.

Hinweise:

1. Wenn Sie entweder für den Konfigurationsparameter **logarchmeth1** oder für **logarchmeth2** einen anderen Wert als OFF festlegen, ist die Datenbank für die aktualisierende Recovery konfiguriert.
2. Wenn Sie die Option **userexit** oder **logretain** für den Konfigurationsparameter **logarchmeth1** verwenden, müssen Sie den Konfigurationsparameter **logarchmeth2** auf OFF setzen.

Wenn Sie den Konfigurationsparameter **logarchmeth2** zum Angeben eines Pfads verwenden, werden Protokolldateien sowohl an diesem Ziel als auch an dem Ziel archiviert, das durch den Datenbankkonfigurationsparameter **logarchmeth1** angegeben wird. Welche Protokolldateien vom Konfigurationsparameter **logarchmeth2** archiviert werden, hängt davon ab, ob Sie auch den Wert für den Konfigurationsparameter **mirrorlogpath** festlegen:

- Wenn Sie den Wert des Konfigurationsparameters **mirrorlogpath** nicht festlegen, archiviert der Konfigurationsparameter **logarchmeth2** Protokolldateien aus dem aktuellen Protokollpfad, wie durch den Konfigurationsparameter **logpath** angegeben.
- Wenn Sie den Wert für den Konfigurationsparameter **mirrorlogpath** festlegen, archiviert der Konfigurationsparameter **logarchmeth2** Protokolldateien aus dem Pfad für die Protokollspiegelung.

logarchopt1 - Optionen für primäres Protokollarchiv

Mit diesem Parameter wird das Feld der Optionen für das primäre Ziel für archivierte Protokolle (falls erforderlich) angegeben.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
NULL [nicht anwendbar]

Einschränkungen
In Umgebungen mit Tivoli Storage Manager (TSM), die für die Unterstützung von Proxy-Knoten konfiguriert sind, sind die Optionen `"-fromnode=nodename"` und `"-fromowner=ownername"` nicht mit der Option `"-asnodename=nodename"` kompatibel und können nicht zusammen verwendet werden. Verwenden Sie die Option `-asnodename` für TSM-Konfigurationen, in denen Proxy-Knoten eingesetzt werden, und die beiden anderen Optionen für andere Typen von TSM-Konfigurationen. Weitere Informationen finden Sie in „Konfigurieren eines Tivoli Storage Manager-Clients”.

logarchopt2 - Optionen für sekundäres Protokollarchiv

Mit diesem Parameter wird das Feld der Optionen für das sekundäre Ziel für archivierte Protokolle (falls erforderlich) angegeben.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
NULL [nicht anwendbar]

Einschränkungen
In Umgebungen mit Tivoli Storage Manager (TSM), die für die Unterstützung von Proxy-Knoten konfiguriert sind, sind die Optionen `"-fromnode=nodename"` und `"-fromowner=ownername"` nicht mit der Option `"-asnodename=nodename"` kompatibel und können nicht zusammen verwendet werden. Verwenden Sie die Option `-asnodename` für TSM-Konfigurationen, in denen Proxy-Knoten eingesetzt werden, und die beiden anderen Optionen für andere Typen von TSM-Konfigurationen. Weitere Informationen finden Sie in „Konfigurieren eines Tivoli Storage Manager-Clients".

logbufsz - Protokollpuffergröße

Mit diesem Parameter kann die Menge an Datenbankzwischenspeicher (der durch den Parameter **dbheap** definiert wird) angegeben werden, die als Puffer für Protokollsätze verwendet werden soll, bevor diese Protokollsätze auf die Festplatte geschrieben werden.

Konfigurationstyp
Datenbank

Parametertyp

- Konfigurierbar
- Nach Member in einer DB2 pureScale-Umgebung konfigurierbar

Standardwert [Bereich]

32-Bit-Plattformen
`256 [4 - 4 096 ]`

64-Bit-Plattformen
`256 [4 - 131 070 ]`

Anmerkung: Der Standardwert unterliegt nach der ursprünglichen Datenbankerstellung Änderungen durch den DB2 Configuration Advisor.

Maßeinheit
Seiten (4 KB)

Protokollsätze werden in folgenden Fällen auf die Festplatte geschrieben:

- Eine Transaktion oder eine Gruppe von Transaktionen wird entsprechend der Angaben für den Konfigurationsparameter **mincommit** festgeschrieben.
- Der Protokollpuffer ist voll.
- Ein anderes internes Ereignis im Datenbankmanager macht das Schreiben auf die Festplatte erforderlich.

Der Wert dieses Parameters muss außerdem kleiner oder gleich dem Wert des Parameters **dbheap** sein. Durch Puffern der Protokollsätze werden E/A-Operationen für die Protokolldateien effektiver, da weniger häufig Schreiboperationen ausgeführt und bei jeder Schreiboperation mehr Protokollsätze auf die Festplatte geschrieben werden.

Empfehlung: Erhöhen Sie den Wert für die Größe dieses Pufferbereichs, wenn umfangreiche Leseaktivitäten auf einer dedizierten Protokollplatte auftreten oder die Festplatte in erheblichem Maße beansprucht wird. Wenn Sie den Wert dieses Parameters erhöhen, sollten Sie auch den Parameter **dbheap** berücksichtigen, da der Protokollpuffer einen Speicherbereich verwendet, der mit dem Parameter **dbheap** gesteuert wird.

Sie können mit dem Datenbanksystemmonitor feststellen, wie häufig der Protokollpuffer vollständig belegt wurde und auf Platte geschrieben werden musste, damit neue Protokollsätze geschrieben werden konnten. Siehe hierzu das Monitorelement **num_log_buffer_full** (Anzahl der vollen Protokollpuffer).

logfilsiz - Protokolldateigröße

Mit diesem Parameter wird die Größe jeder primären und sekundären Protokolldatei definiert. Die Größe dieser Protokolldateien beschränkt die Anzahl der Protokollsätze, die in die Dateien geschrieben werden können, bevor sie voll sind und eine neue Protokolldatei erforderlich wird.

Konfigurationstyp
Datenbank

Parametertyp
Konfigurierbar

Standardwert [Bereich]

UNIX `1000 [4 - 1 048 572]`

Windows
`1000 [4 - 1 048 572]`

Anmerkung: Der Standardwert unterliegt nach der ursprünglichen Datenbankerstellung Änderungen durch den DB2 Configuration Advisor.

Maßeinheit
Seiten (4 KB)

Die Verwendung primärer und sekundärer Protokolldateien sowie die Maßnahmen, die ausgeführt werden, wenn eine Protokolldatei voll ist, sind von der Art der ausgeführten Protokollierung abhängig:

- Umlaufprotokollierung

  Eine primäre Protokolldatei kann wieder verwendet werden, wenn die in ihr aufgezeichneten Änderungen mit COMMIT festgeschrieben wurden. Wenn die Größe der Protokolldatei beschränkt ist und Anwendungen eine große Anzahl von Änderungen an der Datenbank vorgenommen haben, ohne die Änderungen festzuschreiben, kann eine primäre Protokolldatei schnell voll werden. Wenn alle primären Protokolldateien voll sind, ordnet der Datenbankmanager sekundäre Protokolldateien für die neuen Protokollsätze zu.
- Beibehaltung der Protokolle oder Archivprotokollierung

Wenn eine Protokolldatei voll ist, wird sie geschlossen und nicht überschrieben. Wenn die Archivprotokollierung konfiguriert ist, wird die Protokolldatei entsprechend archiviert.

Empfehlung: Die Größe der Protokolldateien muss mit der Anzahl der Protokolldateien abgestimmt werden:

- Der Wert des Parameters **logfilsiz** sollte erhöht werden, wenn an der Datenbank eine große Anzahl von Aktualisierungs-, Lösch- oder Einfügetransaktionen ausgeführt wird, durch die die Protokolldatei sehr schnell voll wird.

  Anmerkung: Der obere Grenzwert für die Größe der Protokolldatei zusammen mit dem oberen Grenzwert für die Anzahl Protokolldateien (**logprimary** + **logsecond**) ergibt einen oberen Grenzwert von 1024 GB für den aktiven Protokollspeicherbereich.

  Eine zu kleine Protokolldatei kann sich auf die Systemleistung auswirken, da das Archivieren alter Protokolldateien, das Zuordnen neuer Protokolldateien sowie das Warten auf verwendbare Protokolldateien zusätzliche Verarbeitungszeit erfordert.
- Der Wert des Parameters **logfilsiz** sollte verringert werden, wenn Plattenspeicherplatz knapp ist, da primäre Protokolldateien im Voraus mit dieser Größe zugeordnet werden.

  Eine zu große Protokolldatei kann Ihre Flexibilität bei der Verwaltung archivierter Protokolldateien bzw. beim Kopieren der Protokolldateien einschränken, da auf einigen Platten vielleicht keine vollständige Protokolldatei gespeichert werden kann.

  Außerdem kann die Größe der Protokolldatei Auswirkungen darauf haben, wie häufig Protokolldateien archiviert werden und wie lange es dauert, eine einzelne Protokolldatei zu archivieren. Diese Faktoren haben Auswirkungen für die Verfügbarkeit der Protokolldateien an der Archivspeicherposition in Disaster-Recoveryszenarios, wenn der Primärserver ausfällt. Mit dem Befehl **ARCHIVE LOG FOR DATABASE** können Protokolldateien abgeschnitten und in kürzeren Zeitabständen archiviert werden, falls erforderlich.

Wenn Sie die Protokollspeicherung verwenden, wird die aktuelle aktive Protokolldatei geschlossen und abgeschnitten, wenn die letzte Anwendung die Verbindung zu einer Datenbank trennt. Für die nächste zur Datenbank hergestellte Verbindung wird die nächste Protokolldatei verwendet. Wenn Sie also die Protokollanforderungen Ihrer gleichzeitig ausgeführten Anwendungen kennen, können Sie vielleicht eine Protokolldateigröße festlegen, durch die nicht zu viel Speicher umsonst zugeordnet wird.

loghead - Erste aktive Protokolldatei

Dieser Parameter gibt den Namen der zurzeit aktiven Protokolldatei an.

Konfigurationstyp
Datenbank

Parametertyp
Informativ

logindexbuild - Erstellte Indexseiten protokollieren

Mit diesem Parameter wird angegeben, ob Operationen zur Erstellung, erneuten Erstellung oder Reorganisation von Indizes protokolliert werden sollen, sodass In-

dizes bei DB2-Operationen zur aktualisierenden Recovery oder bei Prozeduren zum Nachvollziehen von HADR-Protokollen (High Availability Disaster Recovery) wiederhergestellt werden können.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp

- Online konfigurierbar
- Nach Member in einer DB2 pureScale-Umgebung konfigurierbar

Standardwert [Bereich]
Off [On; Off]

logpath - Pfad zu Protokolldateien

Dieser Parameter gibt den aktuellen Pfad an, der für die Protokolldateien verwendet wird.

Konfigurationstyp
Datenbank

Parametertyp
Informativ

Dieser Parameter kann nicht direkt geändert werden, da der Wert vom Datenbankmanager festgelegt wird, wenn eine Änderung am Parameter **newlogpath** wirksam wird.

Der Standardprotokollpfad für ein Member ist ein Verzeichnis innerhalb des globalen Datenbankverzeichnisses. Beispielsweise kann der Standardprotokollpfad bei Verwendung des Instanznamens 'dbinst', des Datenbanknamens 'dbname' und des Benutzernamens 'dbuser' für ein Member wie folgt lauten: `/home/dbuser/dbinst/NODE0000/SQL00001/LOGSTREAM0000`. In einer DB2 pureScale-Umgebung gibt es für jedes Member ein Verzeichnis LOGSTREAM*xxxx*.

logprimary - Anzahl primärer Protokolldateien

Mit diesem Parameter kann die Anzahl der im Voraus zugeordneten primären Protokolldateien festgelegt werden. Die primären Protokolldateien reservieren eine feste Menge Speicher, der den Recoveryprotokollen zugeordnet wird.

Konfigurationstyp
Datenbank

Parametertyp
Konfigurierbar

Standardwert [Bereich]
`3 [ 2 - 256 ]`

Anmerkung: Der Standardwert unterliegt nach der ursprünglichen Datenbankerstellung Änderungen durch den DB2 Configuration Advisor.

Maßeinheit
Zähler

Zuordnung

- Wenn die Datenbank erstellt wird.
- Wenn ein Protokoll an eine andere Speicherposition versetzt wird (dies geschieht, wenn der Parameter **newlogpath** aktualisiert wird).
- Wenn die Datenbank das nächste Mal nach einer Erhöhung des Werts dieses Parameters (**logprimary**) gestartet wird, sofern die Datenbank nicht als HADR-Bereitschaftsdatenbank gestartet wird.
- Wenn eine Protokolldatei inaktiv wird und eine neue Protokolldatei benötigt wird, um zumindest die **logprimary**-Protokolle im primären Protokollpfad beizubehalten (für den Parameter **logarchmeth1** oder den Parameter **logarchmeth2** darf kein anderer Wert als OFF festgelegt werden).
- Wenn der Parameter **logfilsiz** geändert wurde, wird die Größe der Protokolldateien beim nächsten Start der Datenbank geändert, sofern die Datenbank nicht als HADR-Bereitschaftsdatenbank gestartet wird.

Freigabe
Eine Freigabe erfolgt, wenn der Wert für diesen Parameter herabgesetzt wird. Wenn der Wert herabgesetzt wird, werden nicht mehr benötigte Protokolldateien bei der nächsten Verbindung zur Datenbank gelöscht.

Bei der Umlaufprotokollierung werden die primären Protokolldateien nacheinander wiederholt verwendet. Das heißt, wenn ein Protokoll voll ist, wird die nächste primäre Protokolldatei in der Reihenfolge verwendet, wenn sie verfügbar ist. Ein Protokoll wird als verfügbar angesehen, wenn alle UOWs (UOW = Unit of Work, Arbeitseinheit) mit Protokollsätzen in diesem Protokoll mit COMMIT festgeschrieben oder mit ROLLBACK rückgängig gemacht wurden. Wenn die nächste Protokolldatei in der Reihenfolge nicht verfügbar ist, wird eine sekundäre Protokolldatei zugeordnet und verwendet. Es werden solange weitere sekundäre Protokolldateien zugeordnet und verwendet, bis die nächste primäre Protokolldatei in der Reihenfolge verfügbar wird oder der durch den Parameter **logsecond** festgelegte Grenzwert erreicht wird. Sobald diese sekundären Protokolldateien vom Datenbankmanager nicht mehr benötigt werden, wird der für sie verwendete Speicher wieder freigegeben.

Für die Anzahl der primären und sekundären Protokolldateien muss Folgendes gelten:

- Wenn **logsecond** den Wert -1 besitzt, ist **logprimary** <= 256.
- Wenn **logsecond** nicht den wert -1 besitzt, ist (**logprimary** + **logsecond**) <= 256.

Empfehlung: Der Wert, der für diesen Parameter gewählt wird, ist von einer Reihe von Faktoren abhängig, zu denen auch die Art der Protokollierung, die Größe der Protokolldateien und die Art der Verarbeitungsumgebung (z.B. die Länge der Transaktionen und die Häufigkeit der Commits) gehören.

Durch die Erhöhung dieses Werts wird mehr Plattenspeicher für die Protokolle erforderlich, weil die primären Protokolldateien bereits bei der ersten Verbindung zur Datenbank im Voraus zugeordnet werden.

Wenn sich herausstellt, dass häufig sekundäre Protokolldateien zugeordnet werden, kann die Systemleistung möglicherweise durch eine Vergrößerung der Protokolldateien (**logfilsiz**) oder durch eine Erhöhung der Anzahl primärer Protokolldateien verbessert werden.

Bei Datenbanken, auf die nicht oft zugegriffen wird, sollte zur Einsparung von Plattenspeicherplatz dieser Parameter auf den Wert 2 gesetzt werden. Bei Datenbanken, für die die aktualisierende Recovery aktiviert ist, sollte dieser Parameter auf einen größeren Wert gesetzt werden, um die zusätzliche Verarbeitungszeit durch die beinahe sofort erforderliche Zuordnung neuer Protokolle zu vermeiden.

Sie können den Datenbanksystemmonitor zur Ermittlung der geeigneten Größe für die primären Protokolldateien verwenden. Die Beobachtung der folgenden Monitorwerte über einen gewissen Zeitraum hinweg kann sich für die geeignete Einstellung der Parameter als nützlich erweisen, da Durchschnittswerte Ihre tatsächlichen Anforderungen wahrscheinlich besser widerspiegeln.

- **sec_log_used_top** (Maximum des verwendeten sekundären Protokollspeichers)
- **tot_log_used_top** (Maximum des verwendeten Gesamtprotokollspeichers)
- **sec_logs_allocated** (Momentan zugeordnete sekundäre Protokolle)

logsecond - Anzahl sekundärer Protokolldateien

Dieser Parameter gibt die Anzahl der sekundären Protokolldateien an, die (nach Bedarf) erstellt und für Recoveryprotokolle verwendet werden.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
10 [-1; 0 – 254]

Anmerkung: Der Standardwert unterliegt nach der ursprünglichen Datenbankerstellung Änderungen durch den DB2 Configuration Advisor.

Maßeinheit
Zähler

Zuordnung
Nach Bedarf, wenn **logprimary** nicht ausreichend ist (siehe unten).

Freigabe
Wenn der Datenbankmanager feststellt, dass sie nicht mehr erforderlich sind.

Wenn die primären Protokolldateien voll sind, werden die sekundären Protokolldateien (in der Größe **logfilsiz**) nacheinander so zugeordnet, wie sie benötigt werden, und zwar höchstens so viele, wie durch diesen Parameter festgelegt wird. Wenn mehr sekundäre Protokolldateien erforderlich sind als dieser Parameter zulässt, wird ein Fehlercode an die Anwendung zurückgegeben.

Wenn Sie **logsecond** auf -1 setzen, wird die Datenbank mit unbegrenztem Speicherbereich für aktive Protokolle konfiguriert. Für die Größe oder Anzahl der unvollständigen Transaktionen, die in der Datenbank ausgeführt werden, gibt es keine Begrenzung. Auch wenn Sie **logsecond** auf den Wert -1 setzen, verwenden Sie die Konfigurationsparameter **logprimary** und **logfilsiz**, um die Anzahl der Protokolldateien anzugeben, die der Datenbankmanager im Pfad für aktive Protokolldateien behalten soll. Wenn der Datenbankmanager Protokolldaten aus einer Protokolldatei lesen muss, die sich nicht im Pfad für aktive Protokolldateien befin-

det, ruft der Datenbankmanager die Protokolldatei aus dem Archiv in den Pfad für aktive Protokolldateien ab. (Der Datenbankmanager ruft die Dateien in den Überlaufprotokollpfad ab, falls Sie einen solchen Pfad konfiguriert haben.) Nach dem Abrufen der Protokolldatei wird diese Datei vom Datenbankmanager im Pfad für aktive Protokolldateien zwischengespeichert, sodass das Lesen weiterer Protokolldaten aus dieser Datei schneller erfolgen kann. Der Datenbankmanager verwaltet das Abrufen, Zwischenspeichern und Entfernen dieser Protokolldateien nach Bedarf.

Anmerkung: Sie können in DB2 pureScale-Umgebungen keinen unbegrenzten Speicherbereich für aktive Protokolle konfigurieren.

Wenn Ihr Protokollpfad eine Roheinheit ist, müssen Sie den Konfigurationsparameter **overflowlogpath** konfigurieren, um **logsecond** auf den Wert -1 setzen zu können.

Wenn Sie **logsecond** auf den Wert -1 setzen, besteht für die Größe der UOW (Unit of Work, Arbeitseinheit) oder die Anzahl gleichzeitig ablaufender UOWs keine Begrenzung. Allerdings können Rollbacks (sowohl auf Sicherungspunktebene als auch auf UOW-Ebene) viel Zeit in Anspruch nehmen, da Protokolldateien aus dem Archiv abgerufen werden müssen. Aus demselben Grund kann die Recovery nach einem Systemabsturz ebenfalls sehr zeitaufwendig sein. Der Datenbankmanager schreibt eine Nachricht in das Protokoll mit Benachrichtigungen für die Systemverwaltung, um Sie darauf hinzuweisen, wenn die aktuelle Menge aktiver UOWs die primären Protokolldateien überschreitet. Dies weist zugleich darauf hin, dass Rollback- oder Recoveryoperationen nach einem Systemabsturz viel Zeit in Anspruch nehmen können.

Damit der Parameter **logsecond** auf den Wert -1 gesetzt werden kann, muss der Konfigurationsparameter **logarchmeth1** auf einen anderen Wert als OFF oder LOGRETAIN gesetzt werden.

Empfehlung: Verwenden Sie sekundäre Protokolldateien für Datenbanken, die in periodischen Zeitabständen immer wieder große Mengen an Protokollspeicher benötigen. Sekundäre Protokolldateien sollten beispielsweise eingesetzt werden, wenn eine Anwendung, die einmal im Monat ausgeführt wird, mehr Protokollspeicher benötigt, als durch die primären Protokolldateien zur Verfügung gestellt wird. Da sekundäre Protokolldateien keinen permanenten Dateibereich erfordern, sind sie für solche Zwecke gut geeignet.

Wenn die unbegrenzte Protokollierung aktiviert ist (**logsecond** hat den Wert -1), reserviert der Datenbankmanager keinen Speicherbereich für aktive Protokolldateien für Transaktionen, die möglicherweise durch Rollback rückgängig gemachten werden und Protokollsätze schreiben müssen. Wenn während der Rollbackverarbeitung sowohl der Pfad für aktive Protokolldateien als auch das Archivzielverzeichnis voll sind (oder wenn kein Zugriff auf das Archivzielverzeichnis möglich ist), sollte der Konfigurationsparameter **blk_log_dsk_ful** ('Bei voller Protokollplatte blockieren') ebenfalls auf ENABLED gesetzt sein, um Datenbankausfälle zu vermeiden.

max_log - Maximale Protokollgröße pro Transaktion

Dieser Parameter gibt an, ob für den Prozentsatz des primären Protokollspeicherbereichs, der von einer Transaktion belegt werden kann, ein Grenzwert definiert wurde und wie dieser lautet.

Konfigurationstyp
Datenbank

Parametertyp

- Online konfigurierbar
- Nach Member in einer DB2 pureScale-Umgebung konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
`0` [`0 - 100`]

Maßeinheit
Prozent

Wenn der Wert nicht `0` ist, gibt dieser Parameter den Prozentsatz des Speicherbereichs für primäre Protokolle an, der von einer Transaktion belegt werden kann.

Wird der Wert auf `0` gesetzt, besteht keine Begrenzung des Prozentsatzes des gesamten primären Protokollbereichs, den eine Transaktion verbrauchen kann.

Verletzt eine Anwendung die Konfiguration von **max_log**, wird die Anwendung gezwungen, die Verbindung zur Datenbank zu beenden, und die Transaktion wird rückgängig gemacht.

Sie können dieses Verhalten außer Kraft setzen, indem Sie die Registrierdatenbankvariable *DB2_FORCE_APP_ON_MAX_LOG* auf „FALSE" setzen. Dies bewirkt, dass Transaktionen, die die Konfiguration von **max_log** verletzen, fehlschlagen; die Anwendung kann jedoch trotzdem die von vorhergehenden Anweisungen in der UOW ausgeführten Arbeitsschritte festschreiben oder einen Rollback der ausgeführten Arbeitsschritte durchführen, um die UOW rückgängig zu machen.

Dieser Parameter kann zusammen mit dem Konfigurationsparameter **num_log_span** nützlich sein, wenn ein unbegrenzter aktiver Protokollspeicherbereich aktiviert wird. Wenn die Endlosprotokollierung aktiviert ist (d. h., für *logsecond* ist der Wert -1 definiert), werden Transaktionen nicht auf die Obergrenze der Anzahl von Protokolldateien (*logprimary* + *logsecond*) beschränkt. Wenn der Wert von *logprimary* erreicht wurde, beginnt DB2 mit der Archivierung der aktiven Protokolldateien, statt die Transaktion fehlschlagen zu lassen. Dies kann zu Problemen führen, wenn zum Beispiel eine Anwendung eine Transaktion mit langer Ausführungszeit enthält, die nicht festgeschrieben wurde. In einen solchen Fall wächst der Protokollspeicherbereich immer weiter an, was sich negativ auf die Leistung bei einer Recovery nach einem Systemabsturz auswirken könnte. Um dies zu verhindern, können Sie Werte für einen der Konfigurationsparameter **max_log** oder **num_log_span** (oder beide) angeben.

Anmerkung: Die folgenden DB2-Befehle sind von der Begrenzung ausgenommen, die vom Konfigurationsparameter **max_log** festgelegt wird: ARCHIVE LOG, BACKUP DATABASE, LOAD, REORG, RESTORE DATABASE und ROLLFORWARD DATABASE.

maxappls - Maximale Anzahl aktiver Anwendungen

Mit diesem Parameter wird die maximale Anzahl der (sowohl lokalen als auch fernen) Anwendungen angegeben, die gleichzeitig auf eine Datenbank zugreifen können. Da jede Anwendung, die die Verbindung zu einer Datenbank herstellt, die Zuordnung privaten Speichers erforderlich macht, entsteht durch Erhöhung dieses Parameters möglicherweise mehr Speicherbedarf.

Konfigurationstyp
Datenbank

Parametertyp

- Online konfigurierbar
- Nach Member in einer DB2 pureScale-Umgebung konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
`Automatic` [`1 - 60 000`]

Anmerkung: Der Standardwert unterliegt nach der ursprünglichen Datenbankerstellung Änderungen durch den DB2 Configuration Advisor.

Maßeinheit
Zähler

Wenn Sie den Parameter **maxappls** auf `automatic` setzen, ermöglichen Sie dadurch eine beliebige Anzahl verbundener Anwendungen. Der Datenbankmanager ordnet die zur Unterstützung neuer Anwendungen erforderlichen Ressourcen dynamisch zu.

Wenn Sie diesen Parameter nicht auf `automatic` setzen wollen, muss der Wert dieses Parameters größer oder gleich der Summe der verbundenen Anwendungen plus der Anzahl eben dieser Anwendungen sein, die sich gleichzeitig im Prozess eines zweiphasigen Commits oder eines Rollbacks befinden können. Addieren Sie zu dieser Summe die vorausgeschätzte Anzahl unbestätigter Transaktionen, die zu einem gegebenen Zeitpunkt vorhanden sein können.

Wenn eine Anwendung versucht, die Verbindung zu einer Datenbank herzustellen, jedoch der Wert des Parameters **maxappls** bereits erreicht wurde, wird an die Anwendung ein Fehler zurückgegeben, dass die maximale Anzahl von Anwendungen bereits mit der Datenbank verbunden ist.

In einer Umgebung mit partitionierten Datenbanken ist dies die maximale Anzahl von Anwendungen, die gleichzeitig auf einer Datenbankpartition aktiv sein können. Dieser Parameter beschränkt die Anzahl aktiver Anwendungen für eine Datenbankpartition auf einem Datenbankpartitionsserver unabhängig davon, ob der Server der Koordinatorknoten für die Anwendung ist oder nicht. Für den Katalogknoten in einer Umgebung mit partitionierten Datenbanken ist ein höherer Wert für **maxappls** erforderlich als für andere Umgebungstypen, weil in der Umgebung mit partitionierten Datenbanken jede Anwendung eine Verbindung zum Katalogknoten benötigt.

Empfehlung: Wenn der Wert dieses Parameters erhöht wird, ohne dass der Wert des Parameters **maxlocks** verringert oder der Wert des Parameters **locklist** erhöht wird, könnte die maximale Anzahl Sperren für die Datenbank (Parameter **locklist**) früher erreicht werden als die maximale Anzahl Anwendungen, was zu ständigen Problemen durch Sperreneskalation führen kann.

Bis zu einem gewissen Grad wird die maximale Anzahl von Anwendungen auch vom Paramater **max_coordagents** bestimmt. Eine Anwendung kann nur dann eine Verbindung zur Datenbank herstellen, wenn eine verfügbare Verbindung (**maxappls**) sowie ein koordinierender Agent (**max_coordagents**) vorhanden sind.

maxfilop - Maximale Anzahl offener Datenbankdateien pro Datenbank

Dieser Parameter gibt die maximale Anzahl der Dateikennungen an, die für jede Anwendung geöffnet sein können.

Konfigurationstyp
Datenbank

Parametertyp

- Online konfigurierbar
- Nach Member in einer DB2 pureScale-Umgebung konfigurierbar

Weitergabeklasse
Transaktionsgrenzwert

Standardwert [Bereich]

AIX, Sun, HP und Linux (64 Bit)
`61 440 [64 - 61 440]`

Linux (32 Bit)
`30 720 [64 - 30 720]`

Windows (32-Bit)
`32 768 [64 - 32 768]`

Windows (64-Bit)
`65 335 [64 - 65 335]`

Maßeinheit
Zähler

Wenn durch das Öffnen einer Datei dieser Wert überschritten wird, werden einige von dieser Datenbank verwendete Dateien geschlossen. Wenn der Wert für den Parameter **maxfilop** zu klein ist, steigt die Verarbeitungszeit für das Öffnen und Schließen von Dateien übermäßig an und kann die Leistung beeinträchtigen.

Sowohl SMS-Tabellenbereiche als auch DMS-Tabellenbereichsdateicontainer werden bei der Interaktion des Datenbankmanagers mit dem Betriebssystem als Dateien behandelt, sodass Dateikennungen für sie erforderlich sind. In der Regel werden von SMS-Tabellenbereichen vergleichsweise mehr Dateien verwendet, als von DMS-Dateitabellenbereichen Container verwendet werden. Daher wird für diesen Parameter ein höherer Wert benötigt, wenn SMS-Tabellenbereiche verwendet werden, als für DMS-Dateitabellenbereiche erforderlich wäre.

Mithilfe dieses Parameters kann außerdem sichergestellt werden, dass die Gesamtzahl der Dateikennungen, die vom Datenbankmanager verwendet werden, den Grenzwert des Betriebssystems nicht überschreitet, indem die Anzahl der Dateikennungen pro Datenbank auf einen bestimmten Wert gesetzt wird. Die tatsächliche Anzahl ist je nach Anzahl der gleichzeitig aktiven Datenbanken unterschiedlich.

maxlocks - Maximale Anzahl von Sperren vor Eskalation

Mit diesem Parameter wird definiert, wie viel Prozent der Sperrenliste eine Anwendung belegen muss, damit der Datenbankmanager eine Sperreneskalation ausführt.

Konfigurationstyp
Datenbank

Parametertyp

- Online konfigurierbar
- Nach Member in einer DB2 pureScale-Umgebung konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
`Automatic [1 - 100 ]`

Anmerkung: Der Standardwert unterliegt nach der ursprünglichen Datenbankerstellung Änderungen durch den DB2 Configuration Advisor.

Maßeinheit
Prozent

Eine Sperreneskalation ist der Prozess, durch den jeweils mehrere Zeilensperren für eine Tabelle durch eine Tabellensperre ersetzt werden, um die Anzahl der Sperren in der Liste zu verringern. Wenn die Anzahl der Sperren, die von einer Anwendung aktiviert wurden, diesen Prozentwert der Gesamtgröße der Sperrenliste erreicht, wird für die Sperren dieser Anwendung eine Sperreneskalation ausgeführt. Eine Sperreneskalation wird auch dann ausgeführt, wenn in der Sperrenliste kein weiterer Platz mehr verfügbar ist.

Der Datenbankmanager bestimmt die zu eskalierenden Sperren, indem er die Sperrenliste nach der Anwendung durchsucht und die Tabelle mit den meisten gesperrten Zeilen ermittelt. Wenn nach der Ersetzung dieser Sperren durch eine einzige Tabellensperre der Wert für **maxlocks** nicht mehr überschritten wird, wird die Sperreneskalation beendet. Andernfalls wird die Eskalation fortgesetzt, bis der von dieser Anwendung belegte Prozentsatz der Sperrenliste unter den Wert von **maxlocks** gesunken ist. Das Produkt aus dem Wert des Parameters **maxlocks** und dem Wert des Parameters **maxappls** darf nicht kleiner als 100 sein.

Der Wert `AUTOMATIC` für diesen Parameter aktiviert ihn für die automatische Leistungsoptimierung. Dadurch ist die Speicheroptimierungsfunktion in der Lage, die durch diesen Parameter gesteuerte Größe des Hauptspeicherbereichs an geänderte Auslastungsanforderungen je nach Bedarf dynamisch anzupassen. Da die Speicheroptimierungsfunktion Speicherressourcen auf verschiedene Speicherverbraucher verteilt, müssen mindestens zwei Speicherverbraucher für die automatische Optimierung aktiviert sein, damit eine automatische Optimierung erfolgen kann.

Obwohl der Wert des Parameters **locklist** zusammen mit dem Parameter **maxlocks** optimiert werden kann, bewirkt die Inaktivierung der automatischen Optimierung des Parameters **locklist** nicht automatisch die Inaktivierung der automatischen Optimierung des Parameters **maxlocks**. Die Aktivierung der automatischen Optimierung für den Parameter **locklist** führt automatisch zu einer Aktivierung der automatischen Optimierung des Parameters **maxlocks**.

Die automatische Optimierung dieses Konfigurationsparameters findet nur statt, wenn der Speicher mit automatischer Leistungsoptimierung für die Datenbank aktiviert ist (der Konfigurationsparameter **self_tuning_mem** muss auf `ON` gesetzt sein).

Auf allen Plattformen erfordert jede Sperre 128 oder 256 Byte der Sperrenliste, und zwar in Abhängigkeit davon, ob für das Objekt andere Sperren aktiv sind:

- 256 Byte sind erforderlich, um eine Sperre für ein Objekt zu aktivieren, für das keine anderen Sperren aktiv sind.
- 128 Byte sind erforderlich, um eine Sperre für ein Objekt einzutragen, für das bereits eine vorhandene Sperre aktiv ist.

Empfehlung: Mit der folgenden Formel können Sie **maxlocks** so einstellen, dass eine Anwendung das Zweifache der durchschnittlichen Anzahl von Sperren aktivieren kann:

```
maxlocks = 2 * 100 / maxappls
```

Dabei wird 2 verwendet, um das Zweifache der durchschnittlichen Anzahl zu erzielen, und 100 ist der höchste zulässige Prozentwert. Wenn Sie nur wenige Anwendungen haben, die gleichzeitig aktiv sind, können Sie alternativ zu der ersten Formel auch die folgende Formel verwenden:

```
maxlocks = 2 * 100 / (durchschnittliche Anzahl der gleichzeitig aktiven
Anwendungen)
```

Bei der Festlegung von **maxlocks** können Sie eine gleichzeitige Verwendung von **locklist** (Größe der Sperrenliste) in Erwägung ziehen. Die maximale Anzahl der Sperren, die eine Anwendung aktivieren kann, bevor eine Sperreneskalation erfolgt, wird wie folgt berechnet:

- `maxlocks * locklist * 4096 /(100 * 128)`

Dabei ist 4096 die Anzahl der Byte auf einer Seite, 100 ist der höchste zulässige Prozentsatz für **maxlocks** und 128 ist die Anzahl der Byte pro Sperre. Wenn Sie wissen, dass eine Ihrer Anwendungen 1000 Sperren benötigt, und Sie keine Sperreneskalation wünschen, sollten Sie die Werte für **maxlocks** und **locklist** in dieser Formel so wählen, dass das Ergebnis größer als 1000 ist. (Wenn Sie für **maxlocks** 10 und für **locklist** 100 verwenden, ist das Ergebnis der Formel höher als die erforderlichen 1000 Sperren.)

Wenn Sie für **maxlocks** einen zu niedrigen Wert wählen, tritt eine Sperreneskalation auf, obwohl für andere gleichzeitig ausgeführte Anwendungen noch genügend Speicher für Sperren verfügbar ist. Wenn ein zu hoher Wert für **maxlocks** gewählt wird, belegen einige wenige Anwendungen u. U. fast den gesamten Speicher für Sperren, während andere Anwendungen eine Sperreneskalation durchführen müssen. Die Notwendigkeit einer Sperreneskalation geht in diesem Fall zulasten des gleichzeitigen Zugriffs.

Mithilfe des Datenbanksystemmonitors können Sie diesen Konfigurationsparameter verfolgen und seinen Wert optimieren.

min_dec_div_3 - 3 Kommastellen bei Dezimaldivision

Dieser Parameter bietet die Möglichkeit, die Berechnung der Kommastellen bei der Dezimaldivision in SQL rasch zu ändern.

Konfigurationstyp
Datenbank

Parametertyp
Konfigurierbar

Standardwert [Bereich]
No [Yes, No]

Der Datenbankkonfigurationsparameter **min_dec_div_3** ändert die Anzahl der Kommastellen im Ergebnis einer Dezimalrechenoperation, die eine Division umfasst. Der Wert kann auf "Yes" oder "No" gesetzt werden. Der Standardwert für **min_dec_div_3** ist "No". Wenn der Wert "No" ist, wird die Anzahl der Kommastellen als 31-p+s-s' berechnet. Wenn der Wert "Yes" ist, wird die Anzahl der Kommastellen als MAX(3, 31-p+s-s') berechnet. Dies führt dazu, dass das Ergebnis einer Dezimaldivision immer mindestens 3 Kommastellen hat. Die Genauigkeit ist immer 31.

Eine Änderung dieses Datenbankkonfigurationsparameters kann Änderungen bei Anwendungen für vorhandene Datenbanken bedingen. Dazu kann es kommen, wenn die Anzahl der Kommastellen bei Ergebnissen von Dezimaldivisionen von der Änderung dieses Datenbankkonfigurationsparameters betroffen wäre. In der folgenden Liste werden einige mögliche Szenarios gezeigt, die Auswirkungen auf Anwendungen haben können. Betrachten Sie diese Szenarios, bevor Sie den Parameter **min_dec_div_3** auf einem Datenbankserver mit vorhandenen Datenbanken ändern.

- Das Verhalten eines statischen Pakets ändert sich erst, wenn das Paket implizit oder explizit erneut gebunden wird. Zum Beispiel werden nach einer Änderung des Werts von NO in YES die zusätzlichen Kommastellen in den Ergebnissen u. U. erst berücksichtigt, nachdem das Paket erneut gebunden wurde. Für jedes geänderte statische Paket kann mit einem expliziten Befehl **REBIND** eine erneute Bindeoperation erzwungen werden.
- Eine Prüfung auf Integritätsbedingung schließt u. U. einige Werte aus, die zuvor akzeptiert wurden. Solche Zeilen verletzen jetzt die Integritätsbedingung, werden jedoch erst entdeckt, wenn eine der Spalten aktualisiert wird, die in der auf Integritätsbedingung geprüften Zeile enthalten ist, oder die Anweisung SET INTEGRITY mit der Option IMMEDIATE CHECKED verarbeitet wird. Sie können die Prüfung auf eine solche Integritätsbedingung erzwingen, indem Sie eine ALTER TABLE-Anweisung ausführen, um die Integritätsbedingung zu löschen und die Integritätsbedingung dann mit einer ALTER TABLE-Anweisung wieder hinzuzufügen.

Anmerkung: Für **min_dec_div_3** gelten außerdem die folgenden Einschränkungen:

1. Der Befehl **GET DB CFG FOR DBNAME** zeigt die Einstellung **min_dec_div_3** nicht an. Die aktuelle Einstellung lässt sich am besten dadurch ermitteln, dass Sie beobachten, welchen Nebeneffekt das Ergebnis einer Dezimaldivision hat. Betrachten Sie z. B. die folgende Anweisung:
   ```
   VALUES (DEC(1,31,0)/DEC(1,31,5))
   ```
   Wenn diese Anweisung den SQLCODE-Wert SQL0419N zurückgibt, unterstützt die Datenbank **min_dec_div_3** nicht oder der Parameter ist auf "No" gesetzt. Wenn die Anweisung 1.000 zurückgibt, ist **min_dec_div_3** auf "Yes" gesetzt.
2. **min_dec_div_3** erscheint nicht in der Liste der Konfigurationsschlüsselwörter, wenn Sie den folgenden Befehl ausführen: `? UPDATE DB CFG`

mincommit - Anzahl der Gruppencommits

Mit diesem Parameter können Sie das Schreiben von Protokollsätzen auf die Platte verzögern, bis eine Mindestanzahl von Commitoperationen durchgeführt wurde. Dies trägt zum Verringern der Verarbeitungszeit bei, die der Datenbankmanager beim Schreiben von Protokollsätzen benötigt.

Wichtig: Dieser Parameter gilt in Version 10.1 als veraltet und wird möglicherweise in einem zukünftigen Release entfernt. Der Parameter kann in Releases vor Version 10.1 weiter verwendet werden. In Version 10.1 und neueren Releases wird der für diesen Konfigurationsparameter angegebene Wert ignoriert.

Konfigurationstyp
Datenbank

Parametertyp

- Online konfigurierbar
- Nach Member in einer DB2 pureScale-Umgebung konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
1 [1 – 25]

Anmerkung: Der Standardwert unterliegt nach der ursprünglichen Datenbankerstellung Änderungen durch den DB2 Configuration Advisor.

Maßeinheit
Zähler

Diese Verzögerung führt zu einer Leistungssteigerung, wenn mehrere Anwendungen für eine Datenbank ausgeführt und von den Anwendungen viele Commitoperationen innerhalb sehr kurzer Zeit angefordert werden.

Diese Gruppierung von Commits wird nur dann ausgeführt, wenn der Wert dieses Parameters größer als eins und die Anzahl der Anwendungen, die mit der Datenbank verbunden sind, größer oder gleich dem Wert dieses Parameters ist. Wenn die Gruppierung von Commits ausgeführt wird, können Commitanforderungen von Anwendungen so lange zurückgehalten werden, bis entweder eine Sekunde vergangen oder die Anzahl der Commitanforderungen gleich dem Wert dieses Parameters ist.

Dieser Parameter sollte nur in kleinen Schritten, zum Beispiel um den Wert 1, erhöht werden. Darüber hinaus sollten Sie mithilfe von Tests mit mehreren Benutzern sicherstellen, dass die Erhöhung des Werts dieses Parameters die erwarteten Ergebnisse liefert. Ein zu hoher Wert für diesen Parameter kann sich negativ auf die Antwortzeit der Anwendungen auswirken.

Am Wert dieses Parameters vorgenommene Änderungen werden sofort wirksam. Sie brauchen nicht abzuwarten, bis alle Anwendungen von der Datenbank getrennt wurden.

Empfehlung: Es empfiehlt sich, diesen Parameter auf den Standardwert 1 zu setzen.

Sie können die Anzahl der Transaktionen pro Sekunde ermitteln und diesen Parameter so anpassen, dass die Höchstanzahl von Transaktionen pro Sekunde (oder ein großer Prozentsatz davon) von diesem Parameter berücksichtigt wird. Durch die Berücksichtigung der Spitzenaktivitäten wird die Verarbeitungszeit beim Schreiben von Protokollsätzen in Phasen mit hohem Transaktionsaufkommen minimiert.

Wenn der Wert des Parameters **mincommit** erhöht wird, kann es zudem notwendig werden, den Wert des Parameters **logbufsz** zu erhöhen, um zu vermeiden, dass ein voller Protokollpuffer eine Schreiboperation während dieser Phasen mit hohem Transaktionsaufkommen erzwingt. In diesem Fall sollte der Wert für den Parameter **logbufsz** nach folgender Formel festgelegt werden:

```
mincommit * (durchschnittlicher Protokollbereich für eine Transaktion)
```

Sie können den Datenbanksystemmonitor folgendermaßen zur Optimierung des Werts dieses Parameters verwenden:

- Berechnen der Höchstanzahl von Transaktionen pro Sekunde:

Durch das Erstellen von Monitorstichproben über einen typischen Arbeitstag hinweg können Sie die Phasen mit hohem Transaktionsaufkommen ermitteln. Sie können die Gesamtanzahl der Transaktionen durch Addieren der Werte für folgende Monitorelemente berechnen:

- **commit_sql_stmts** (versuchte COMMIT-Anweisungen)
- **rollback_sql_stmts** (versuchte ROLLBACK-Anweisungen)

Anhand dieser Stichproben und der verfügbaren Zeitmarken können Sie die Anzahl der Transaktionen pro Sekunde berechnen.

- Berechnen des pro Transaktion verwendeten Protokollspeicherbereichs:

  Mithilfe von Stichproben über einen gewissen Zeitraum hinweg und für eine Anzahl von Transaktionen können Sie den durchschnittlich verwendeten Protokollspeicherbereich mit dem folgenden Monitorelement berechnen:

  - **log_space_used** (verwendeter UOW-Protokollspeicher)

mirrorlogpath - Pfad für Protokollspiegelung

Mit diesem Parameter wird eine Zeichenfolge von bis zu 242 Byte für den Pfad für die Protokollspiegelung angegeben. Die Zeichenfolge muss auf einen vollständig qualifizierten Pfadnamen verweisen.

Konfigurationstyp
Datenbank

Parametertyp
Konfigurierbar

Standardwert [Bereich]
Null [gültiger Pfad oder gültige Einheit]

Sowohl in DB2 pureScale-Umgebungen als auch in DB2 Enterprise Server Edition-Umgebungen werden die Datenbankpartitionsnummer und eine Protokolldatenstrom-ID automatisch an den Pfad angehängt. Beispiel: `/home/dbuser/dblogs/NODE0000/LOGSTREAM0000/`.

Wenn Sie den Wert für den Konfigurationsparameter **mirrorlogpath** festlegen, erstellt das DB2-Datenbanksystem aktive Protokolldateien sowohl im Protokollpfad als auch im Pfad für die Protokollspiegelung. Alle Protokolldaten werden in beide Pfade geschrieben. Der Pfad für die Protokollspiegelung enthält Kopien der aktiven Protokolldateien, sodass die Datenbank im Fall eines Plattenfehlers oder eines Benutzerfehlers, durch den aktive Protokolldateien in einem der Pfade beschädigt werden, weiterhin funktioniert.

In einer DB2 pureScale-Umgebung verarbeitet das erste Member, das eine Verbindung zur Datenbank herstellt oder sie aktiviert, die Änderungen am Wert dieses Protokollpfadparameters. Der DB2-Datenbankmanager stellt sicher, dass der Pfad vorhanden ist und dass Lese- und Schreibzugriff für diesen Pfad besteht. Der Datenbankmanager versucht darüber hinaus, memberspezifische Unterverzeichnisse für die Protokolldateien zu erstellen. Wenn eine dieser Operationen fehlschlägt, weist der DB2-Datenbankmanager den angegebenen Pfad zurück und schaltet die Datenbank mit dem alten Pfad online. Wenn der Datenbankmanager den angegebenen Pfad akzeptiert, wird der neue Wert auf jedes Member repliziert. Wenn ein Member beim Umschalten auf den neuen Pfad fehlschlägt, schlagen nachfolgende Versuche zum Aktivieren oder zum Herstellen einer Verbindung zu ihm mit der Nachricht SQL5099N fehl. Alle Member müssen denselben Protokollpfad verwenden.

Wenn Sie den Pfad für die Protokollspiegelung ändern, können sich im alten Pfad für die Protokollspiegelung immer noch Protokolldateien befinden. Diese Protokolldateien wurden eventuell nicht archiviert, sodass Sie sie möglicherweise manuell archivieren müssen. Wenn Sie zudem für die zugehörige Datenbank eine Replikation ausführen, benötigt die Replikation eventuell weiterhin die vor der Protokollpfadänderung vorhandenen Protokolldateien. Wenn die Datenbank für die Verwendung der Protokollarchivierung konfiguriert ist und alle Protokolldateien entweder automatisch vom DB2-Datenbanksystem oder von Ihnen manuell archiviert wurden, kann das DB2-Datenbanksystem die Protokolldateien zum Beenden des Replikationsprozesses abrufen. Andernfalls können Sie die Dateien aus dem alten Pfad für die Protokollspiegelung in den neuen Pfad für die Protokollspiegelung kopieren.

Wenn der Konfigurationsparameter **logpath** oder **newlogpath** eine Roheinheit als Position zum Speichern der Protokolldateien angibt, ist die Protokollspiegelung, die durch den Konfigurationsparameter **mirrorlogpath** angegeben wird, nicht zulässig. Wenn der Konfigurationsparameter **logpath** oder **newlogpath** einen Dateipfad als Position zum Speichern der Protokolldateien angibt, ist die Protokollspiegelung zulässig, und der Konfigurationsparameter **mirrorlogpath** muss einen Dateipfad angeben.

Der Konfigurationsparameter **mirrorlogpath** hat Auswirkungen auf das Verhalten für die Protokollarchivierung, wenn Sie zusätzlich den Konfigurationsparameter **logarchmeth2** festlegen. Wenn Sie sowohl einen Wert für den Konfigurationsparameter **mirrorlogpath** als auch für **logarchmeth2** angeben, archiviert der Konfigurationsparameter **logarchmeth2** Protokolldateien aus dem Pfad für Protokollspiegelung und nicht zusätzliche Kopien der Protokolldateien im aktiven Protokollpfad. Sie können dieses Verhalten der Protokollarchivierung nutzen, um die Ausfallsicherheit bei der aktualisierenden Recovery zu verbessern. Selbst wenn eine primäre Protokolldatei vor dem Archivieren beschädigt wurde, steht möglicherweise im Pfad für Protokollspiegelung dennoch eine verwendbare archivierte Protokolldatei zur Verfügung, mit der eine Datenbankrecovery fortgesetzt werden kann.

Empfehlung:

- Wie die Protokolldateien sollten sich auch die Spiegelprotokolldateien auf einer physischen Platte mit nur geringer E/A befinden.
- Es wird dringend empfohlen, dass sich der Pfad für die Protokollspiegelung und der primäre Protokollpfad auf unterschiedlichen Einheiten befinden.

Mit dem Datenbanksystemmonitor können Sie die Anzahl der E/A-Operationen für die Datenbankprotokollierung verfolgen. Die folgenden Datenelemente geben das Volumen der E/A-Aktivitäten für die Datenbankprotokollierung zurück.

log_reads
Die Anzahl der gelesenen Protokollseiten.

log_writes
Die Anzahl der geschriebenen Protokollseiten.

Sie können ein Tool zur Betriebssystemüberwachung verwenden, um Informationen zu anderen Platten-E/A-Aktivitäten zu sammeln. Anschließend können Sie beide Arten von E/A-Aktivitäten vergleichen.

mon_act_metrics - Aktivitätsmesswerte überwachen (Konfigurationsparameter)

Dieser Parameter steuert die Erfassung von Aktivitätsmesswerten in der gesamten Datenbank und wirkt sich auf die Aktivitäten aus, die von Verbindungen übergeben werden, die einer beliebigen der DB2-Auslastungsdefinitionen (Workloaddefinition) zugeordnet sind.

Konfigurationstyp
Datenbank

Parametertyp
- Online konfigurierbar

Standardwert [Bereich]
BASE [NONE, BASE, EXTENDED]

Hinweis zu Upgrades
Bei Datenbanken, die vor V9.7 erstellt und auf V9.7 oder eine höhere Version aktualisiert wurden, ist der Parameter **mon_act_metrics** standardmäßig mit NONE definiert.

Wenn Sie diesen Konfigurationsparameter auf den Wert BASE setzen, werden alle Messwerte, die durch die folgenden Schnittstellen zurückgemeldet werden, für alle Aktivitäten erfasst, die auf dem Datenserver ausgeführt werden, und zwar ungeachtet der DB2-Auslastung, der die Verbindung, die die Aktivität übergibt, zugeordnet ist:

- MON_GET_PKG_CACHE_STMT_DETAILS
- MON_GET_ACTIVITY_DETAILS
- MON_GET_PKG_CACHE_STMT
- Ereignismonitor für den Paketcache
- Aktivitätsereignismonitor

Wenn Sie diesen Konfigurationsparameter auf den Wert EXTENDED setzen, werden dieselben Messwerte erfasst wie mit der Einstellung BASE. Darüber hinaus werden die Werte, die für die folgenden Monitorelemente dokumentiert werden, mit erweiterter Granularität ermittelt:

- **total_section_time**
- **total_section_proc_time**
- **total_routine_user_code_time**
- **total_routine_user_code_proc_time**
- **total_routine_time**

Informationen zu den Auswirkungen der Einstellung EXTENDED auf diese Monitorelemente finden Sie in den detaillierten Beschreibungen der jeweiligen Monitorelemente.

Wenn Sie diesen Konfigurationsparameter auf den Wert NONE setzen, werden die Messwerte, die durch die oben aufgeführten Schnittstellen zurückgemeldet werden, nur für die Untergruppe von Aktivitäten erfasst, die von einer Verbindung übergeben werden, die einer DB2-Auslastung zugeordnet ist, deren Klausel COLLECT ACTIVITY METRICS auf den Wert BASE gesetzt wurde.

mon_deadlock - Deadlocks überwachen (Konfigurationsparameter)

Dieser Parameter steuert die Generierung von Deadlock-Ereignissen auf der Datenbankebene für den Sperrereignismonitor.

Konfigurationstyp
Datenbank

Parametertyp

- Online konfigurierbar

Standardwert [Bereich]
`WITHOUT_HIST` [`NONE`, `WITHOUT_HIST`, `HISTORY`, `HIST_AND_VALUES`]

Wenn dieser Parameter auf `NONE` gesetzt wird, werden keine Deadlock-Ereignisse generiert.

Wenn Sie den Parameter auf `WITHOUT_HIST` setzen, werden Daten zu Deadlock-Ereignissen an alle aktiven Sperrereignismonitore gesendet, wenn das Deadlock-Ereignis auftritt. Die für Deadlock-Warteereignisse erfassten Daten enthalten kein Protokoll für Aktivitäten, die von der auf das Deadlock wartenden Anwendung ausgeführt werden.

Wenn Sie den Parameter auf `HISTORY` setzen, enthält das Ereignis vom Typ 'Wartestatus für Sperre' ein Protokoll der Aktivitäten, die von der auf die Sperre wartenden Anwendung ausgeführt wurden. Dieses Protokoll enthält nur Aktivitäten, die in der aktuellen Transaktion für die Anwendung ausgeführt wurden und es zeichnet nur die neuesten ausgeführten Aktivitäten auf, wenn die maximale Größe erreicht ist. Die Standardprotokollgröße ist 250. Die Protokollgröße kann mit der Registrierdatenbankvariablen **DB2_MAX_INACT_STMTS** konfiguriert werden.

Wenn Sie den Parameterwert auf `HIST_AND_VALUES` setzen, enthält das aufgezeichnete Aktivitätsprotokoll für das Ereignis vom Typ 'Wartestatus für Sperre' die Eingabedatenwerte für die betreffenden Aktivitäten. Diese Daten enthalten keine LOB-Daten, LONG VARCHAR-Daten für den Anfang von Änderungen, LONG VARGRAPHIC-Daten, Daten strukturierter Typen für das Ende von Änderungen oder XML-Daten.

Dieser Parameter steuert die Generierung von Deadlock-Ereignissen auf der Datenbankebene für den Sperrereignismonitor. Der Parameter **mon_deadlock** legt fest, ob ein Deadlock-Warteereignis vom Sperrereignismonitor aufgezeichnet wird, wenn ein Deadlock auftritt.

Der Parameterwert für **mon_deadlock** stellt eine Mindesterfassungsebene dar, die für alle DB2-Anwendungen aktiviert ist. Wenn einzelne DB2-Auslastungen eine höhere Erfassungsstufe angeben als die Konfigurationsparameter, wird die DB2-Auslastungseinstellung anstelle des Konfigurationsparameterwerts verwendet.

Zur Erfassung von Deadlocks mit dem Sperrereignismonitor aktivieren Sie die Deadlock-Erfassung auf der Datenbankebene, da sich Operationen, die auf Sperren warten, und Operationen, die Sperren aktiviert haben, in verschiedenen Auslastungen aktiv sein können. Die Ebene der von einem Deadlock erfassten Daten kann einzeln auf der Auslastungsebene gesteuert oder durch diesen Parameter auf der Datenbankebene festgelegt werden.

mon_locktimeout - Überschreitungen des Sperrzeitlimits überwachen (Konfigurationsparameter)

Dieser Parameter steuert die Generierung von Ereignissen bei Überschreitungen des Sperrzeitlimits auf der Datenbankebene für den Sperrereignismonitor und wirkt sich auf alle DB2-Auslastungsdefinitionen (Workloaddefinitionen) aus.

Konfigurationstyp
Datenbank

Parametertyp

- Online konfigurierbar

Standardwert [Mindesterfassungsebene, die für alle Auslastungen bzw. Serviceklassen in der Datenbank aktiviert ist]
NONE [NONE, WITHOUT_HIST, HISTORY, HIST_AND_VALUES]

Wenn Sie diesen Parameter auf NONE setzen, werden Ereignisse für die Überschreitung des Sperrzeitlimits nur generiert, wenn die Ereignisaufzeichnung für Sperrzeitlimitüberschreitungen für DB2-Workload-Objekte mithilfe der Klausel COLLECT LOCK TIMEOUT DATA aktiviert wurde.

Wenn Sie den Parameter auf WITHOUT_HIST setzen, werden Daten zu Sperrereignissen an alle aktiven Sperrereignismonitore gesendet, wenn das Sperrereignis auftritt. Die für die Sperrzeitlimitüberschreitung aufgezeichneten Ereignisdaten enthalten kein Protokoll der Aktivitäten, die von der auf die Sperre wartenden Anwendung ausgeführt wurden.

Wenn Sie den Parameter auf HISTORY setzen enthält das Ereignis für Sperrzeitlimitüberschreitung ein Protokoll der Aktivitäten, die von der auf die Sperre wartenden Anwendung ausgeführt wurden. Dieses Protokoll enthält nur Aktivitäten, die in der aktuellen Transaktion für die Anwendung ausgeführt wurden und es zeichnet nur die neuesten ausgeführten Aktivitäten auf, wenn die maximale Größe erreicht ist. Die Standardprotokollgröße ist 250. Die Protokollgröße kann mit der Registrierdatenbankvariablen **DB2_MAX_INACT_STMTS** konfiguriert werden.

Wenn Sie den Parameterwert auf HIST_AND_VALUES setzen, enthält das aufgezeichnete Aktivitätsprotokoll für die Sperrzeitlimitüberschreitung die Eingabedatenwerte für die betreffenden Aktivitäten. Diese Daten enthalten keine LOB-Daten, LONG VARCHAR-Daten für den Anfang von Änderungen, LONG VARGRAPHIC-Daten, Daten strukturierter Typen für das Ende von Änderungen oder XML-Daten.

Dieser Parameter steuert die Ereignisaufzeichnung für Sperrzeitlimitüberschreitungen auf Datenbankebene für den Sperrereignismonitor. Der Parameter **mon_locktimeout** legt fest, ob ein Sperrzeitlimitüberschreitungsereignis vom Sperrereignismonitor aufgezeichnet wird, wenn eine auf eine Sperre wartende Anwendung das Zeitlimit überschreitet.

Der Parameterwert für **mon_locktimeout** stellt eine Mindesterfassungsebene dar, die für alle DB2-Anwendungen aktiviert ist. Die Erfassung von Sperrzeitlimitüberschreitungsereignissen kann für eine Untermenge von DB2-Anwendungen mit der Klausel COLLECT LOCK TIMEOUT DATA für ein DB2-Workload-Objekt aktiviert werden, anstatt mit dem Parameter **mon_locktimeout**. Wenn einzelne DB2-Auslastungen eine höhere Erfassungsstufe angeben als die Konfigurationsparameter, wird die DB2-Auslastungseinstellung anstelle des Konfigurationsparameterwerts verwendet.

mon_lockwait - Sperrwartestatus überwachen (Konfigurationsparameter)

Dieser Parameter steuert die Generierung von Sperrwarteereignissen auf der Datenbankebene für den Sperrereignismonitor.

Konfigurationstyp
Datenbank

Parametertyp

- Online konfigurierbar

Standardwert [Bereich]
NONE [NONE, WITHOUT_HIST, HISTORY, HIST_AND_VALUES]

Wenn Sie diesen Parameter auf NONE setzen, werden Ereignisse vom Typ 'Wartestatus für Sperre' nur generiert, wenn die Aufzeichnung für Ereignisse vom Typ 'Wartestatus für Sperren' für DB2-Workload-Objekte mithilfe der Klausel COLLECT LOCK WAIT DATA aktiviert wurde.

Wenn Sie den Parameter auf WITHOUT_HIST setzen, werden Daten zu Sperrereignissen an alle aktiven Sperrereignismonitore gesendet, wenn das Sperrereignis auftritt. Die für Ereignisse vom Typ 'Wartestatus für Sperre' erfassten Daten enthalten kein Protokoll der Aktivitäten, die von der auf die Sperre wartenden Anwendung ausgeführt wurden.

Wenn Sie den Parameter auf HISTORY setzen, enthält das Ereignis vom Typ 'Wartestatus für Sperre' ein Protokoll der Aktivitäten, die von der auf die Sperre wartenden Anwendung ausgeführt wurden. Dieses Protokoll enthält nur Aktivitäten, die in der aktuellen Transaktion für die Anwendung ausgeführt wurden und es zeichnet nur die neuesten ausgeführten Aktivitäten auf, wenn die maximale Größe erreicht ist. Die Standardprotokollgröße ist 250. Die Protokollgröße kann mit der Registrierdatenbankvariablen **DB2_MAX_INACT_STMTS** konfiguriert werden.

Wenn Sie den Parameterwert auf HIST_AND_VALUES setzen, enthält das aufgezeichnete Aktivitätsprotokoll für das Ereignis vom Typ 'Wartestatus für Sperre' die Eingabedatenwerte für die betreffenden Aktivitäten. Diese Daten enthalten keine LOB-Daten, LONG VARCHAR-Daten für den Anfang von Änderungen, LONG VARGRAPHIC-Daten, Daten strukturierter Typen für das Ende von Änderungen oder XML-Daten.

Der Parameterwert für **mon_lockwait** stellt eine Mindesterfassungsebene dar, die für alle DB2-Anwendungen aktiviert ist. Die Erfassung der Ereignisse vom Typ 'Wartestatus für Sperre' kann für eine Untermenge von DB2-Anwendungen mit der Klausel COLLECT LOCK WAIT DATA für ein DB2-Workload-Objekt aktiviert werden, anstatt mit dem Parameter **mon_lockwait**. Wenn einzelne DB2-Auslastungen eine höhere Erfassungsstufe angeben als die Konfigurationsparameter, wird die DB2-Auslastungseinstellung anstelle des Konfigurationsparameterwerts verwendet.

Dieser Parameter steuert die Erfassung von Ereignissen des Typs 'Wartestatus für Sperre' auf Datenbankebene für den Sperrereignismonitor. Der Konfigurationsparameter **mon_lockwait** wird in Kombination mit dem Konfigurationsparameter **mon_lw_thresh** verwendet. Der Parameter **mon_lockwait** legt fest, ob ein Ereignis vom Typ 'Wartestatus für Sperre' vom Sperrereignismonitor aufgezeichnet wird, wenn eine Anwendung länger als **mon_lw_thresh** Mikrosekunden auf eine Sperre wartet.

mon_lw_thresh - Schwellenwert für Sperrenwartestatus überwachen (Konfigurationsparameter)

Dieser Parameter steuert die Zeitdauer, die im Sperrenwartestatus abgewartet wird, bevor ein Ereignis für **mon_lockwait** generiert wird.

Konfigurationstyp
Datenbank

Parametertyp

- Online konfigurierbar

Standardwert [Bereich]
5000000 [1000 ... MAX_INT]

Hinweis zu Upgrades
Bei Datenbanken, die vor V9.7 erstellt und auf V9.7 oder eine höhere Version aktualisiert wurden, ist der Parameter **mon_lw_thresh** standardmäßig mit 4294967295 definiert.

Maßeinheit
Mikrosekunden

Wenn dieser Parameter auf der Datenbankebene und auf der Auslastungsebene (Workloadebene) definiert ist, wird die kürzere der beiden konfigurierten Zeiten für die jeweilige Auslastung betrachtet.

mon_lck_msg_lvl - Benachrichtigungen zu Sperrereignissen überwachen (Konfigurationsparameter)

Dieser Parameter steuert die Protokollierung von Nachrichten im Protokoll mit Benachrichtigungen für die Systemverwaltung, wenn Ereignisse wie Sperrenzeitlimitüberschreitungen, Deadlocks und Sperreneskalationen auftreten.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
1 [0 - 3]

Bei Auftreten von Ereignissen durch Überschreitungen von Zeitlimits, Deadlocks und Sperreneskalationen können Nachrichten im Protokoll mit Benachrichtigungen für die Systemverwaltung protokolliert werden, wenn dieser Datenbankkonfigurationsparameter auf einen Wert gesetzt wird, der der gewünschten Benachrichtigungsebene entspricht. Die folgende Liste skizziert die Benachrichtigungsebenen, die festgelegt werden können:

0 Ebene 0: Keine Benachrichtigungen über Sperreneskalationen, Deadlocks und Überschreitungen des Sperrzeitlimits

1 Ebene 1: Benachrichtigungen über Sperreneskalationen

2 Ebene 2: Benachrichtigungen über Sperreneskalationen und Deadlocks

3 Ebene 3: Benachrichtigungen über Sperreneskalationen, Deadlocks und Überschreitungen des Sperrzeitlimits

Die Standardeinstellung dieses Datenbankkonfigurationsparameters für die Benachrichtigungsebene ist 1.

mon_obj_metrics - Datenobjektmesswerte überwachen (Konfigurationsparameter)

Dieser Parameter steuert die Erfassung von Datenobjektmesswerten in einer gesamten Datenbank.

Konfigurationstyp
Datenbank

Parametertyp

- Online konfigurierbar

Standardwert [Bereich]
BASE [NONE, BASE]

Hinweis zu Upgrades
Bei Datenbanken, die vor V9.7 erstellt und auf V9.7 oder eine höhere Version aktualisiert wurden, ist der Parameter **mon_obj_metrics** standardmäßig mit NONE definiert.

Legen Sie für diesen Konfigurationsparameter die Einstellung BASE fest, um alle Messwerte zu erfassen, die von den folgenden Tabellenfunktionen zurückgemeldet werden:

- MON_GET_BUFFERPOOL
- MON_GET_CONTAINER
- MON_GET_TABLESPACE

Legen Sie für diesen Konfigurationsparameter die Einstellung EXTENDED fest, um zusätzliche Messwerte zu erfassen, die von den folgenden Tabellenfunktionen zurückgemeldet werden:

Tabelle 138. Über die Option EXTENDED zurückgegebene Messwerte

Tabellenfunktion	Gemeldete Messwerte
MON_GET_INDEX	object_index_gbp_indep_pages_found_in_lbpobject_index_gbp_invalid_pagesobject_index_gbp_l_readsobject_index_gbp_p_readsobject_index_l_readsobject_index_lbp_pages_foundobject_index_p_reads
MON_GET_INDEX_USAGE_LIST	object_index_gbp_indep_pages_found_in_lbpobject_index_gbp_invalid_pagesobject_index_gbp_l_readsobject_index_gbp_p_readsobject_index_l_readsobject_index_lbp_pages_foundobject_index_p_reads
MON_GET_TABLE	direct_read_reqsdirect_readsdirect_write_reqsdirect_writeslock_escalslock_escals_globallock_wait_timelock_wait_time_globallock_waitslock_waits_globalobject_data_gbp_indep_pages_found_in_lbpobject_data_gbp_invalid_pagesobject_data_gbp_l_readsobject_data_gbp_p_readsobject_data_l_readsobject_data_lbp_pages_foundobject_data_p_readsobject_xda_gbp_indep_pages_found_in_lbpobject_xda_gbp_invalid_pagesobject_xda_gbp_l_readsobject_xda_gbp_p_readsobject_xda_l_readsobject_xda_lbp_pages_foundobject_xda_p_reads

Tabelle 138. Über die Option EXTENDED zurückgegebene Messwerte (Forts.)

Tabellenfunktion	Gemeldete Messwerte
MON_GET_TABLE_USAGE_LIST	direct_read_reqsdirect_readsdirect_write_ reqsdirect_writeslock_escalslock_escals_ globallock_wait_timelock_wait_time_global lock_waitslock_waits_globalobject_data_gbp_ indep_pages_found_in_lbpobject_data_gbp_ invalid_pagesobject_data_gbp_l_readsobject_ data_gbp_p_readsobject_data_l_readsobject_ data_lbp_pages_foundobject_data_p_ readsobject_xda_gbp_indep_pages_found_ in_lbpobject_xda_gbp_invalid_pagesobject_ xda_gbp_l_readsobject_xda_gbp_p_ readsobject_xda_l_readsobject_xda_lbp_ pages_foundobject_xda_p_readsoverflow_ accessesoverflow_createsrows_deletedrows_ insertedrows_readrows_updated

Wenn Sie diesen Konfigurationsparameter auf den Wert NONE setzen, werden die von den zuvor erwähnten Tabellenfunktionen zurückgemeldeten Messwerte nicht erfasst.

mon_pkglist_sz - Paketlistengröße überwachen (Konfigurationsparameter)

Dieser Parameter steuert die maximale Anzahl von Einträgen, die in der Paketliste pro UOW als durch den UOW-Ereignismonitor erfasst enthalten sein können.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Nächste UOW (Unit of Work)

Standardwert [Bereich]
32 [0 - 1024]

Maßeinheit
Anzahl von Einträgen in der Paketliste

Die Paketliste erhält maximal die Größe, die durch den Wert für diesen Datenbankkonfigurationsparameter angegeben wird. Die Größe der Paketliste wird beim Start der UOW ermittelt. Änderungen an der Paketlistengröße werden erst bei der nächsten UOW berücksichtigt. Die Standardgröße für die Paketliste beträgt 32 Einträge.

mon_req_metrics - Anforderungsmesswerte überwachen (Konfigurationsparameter)

Dieser Parameter steuert die Erfassung von Anforderungsmesswerten für die gesamte Datenbank und wirkt sich auf Anforderungen in allen DB2-Serviceklassen aus.

Konfigurationstyp
Datenbank

Parametertyp

- Online konfigurierbar

Standardwert [Bereich]

BASE [NONE, BASE, EXTENDED]

Hinweis zu Upgrades

Bei Datenbanken, die vor V9.7 erstellt und auf V9.7 oder eine höhere Version aktualisiert wurden, ist der Parameter **mon_req_metrics** standardmäßig mit NONE definiert.

Wenn Sie diesen Konfigurationsparameter auf den Wert BASE setzen, werden alle Messwerte, die durch die folgenden Schnittstellen zurückgemeldet werden, für alle Anforderungen erfasst, die auf dem Datenserver ausgeführt werden, und zwar ungeachtet der DB2-Serviceklasse der Anforderungen:

- MON_GET_UNIT_OF_WORK
- MON_GET_UNIT_OF_WORK_DETAILS
- MON_GET_CONNECTION
- MON_GET_CONNECTION_DETAILS
- MON_GET_SERVICE_SUBCLASS
- MON_GET_SERVICE_SUBCLASS_DETAILS
- MON_GET_WORKLOAD
- MON_GET_WORKLOAD_DETAILS
- Statistikereignismonitor (Monitorelement DETAILS_XML in den logischen Datengruppen für 'event_wlstats' und 'event_scstats')
- UOW-Ereignismonitor

Wenn Sie diesen Konfigurationsparameter auf den Wert EXTENDED setzen, werden dieselben Messwerte erfasst wie mit der Einstellung BASE. Darüber hinaus werden die Werte, die für die folgenden Monitorelemente dokumentiert werden, mit erweiterter Granularität ermittelt:

- **total_section_time**
- **total_section_proc_time**
- **total_routine_user_code_time**
- **total_routine_user_code_proc_time**
- **total_routine_time**

Informationen zu den Auswirkungen der Einstellung EXTENDED auf diese Monitorelemente finden Sie in den detaillierten Beschreibungen der jeweiligen Monitorelemente.

Wenn Sie diesen Konfigurationsparameter auf den Wert NONE setzen, werden die Messwerte, die durch die oben aufgeführten Schnittstellen zurückgemeldet werden, nur für die Untergruppe von Anforderungen erfasst, die in einer DB2-Serviceklasse ausgeführt werden, für deren Servicesuperklasse die Klausel COLLECT REQUEST METRICS auf den Wert BASE gesetzt ist.

mon_uow_data - UOW-Ereignisse überwachen (Konfigurationsparameter)

Dieser Parameter steuert die Generierung von UOW-Ereignissen (UOW, Unit of Work) auf der Datenbankebene für den UOW-Ereignismonitor und wirkt sich auf UOWs auf dem Datenserver aus. Er ist ein übergeordneter Parameter für die Konfigurationsparameter **mon_uow_execlist** und **mon_uow_pkglist**.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Nächste UOW (Unit of Work)

Standardwert [Bereich]
NONE [NONE, BASE]

Gültige Werte für diesen Parameter sind wie folgt:

NONE Jegliche Datenerfassung für einen UOW-Ereignismonitor ist inaktiviert.

BASE Die Erfassung von Basisdaten für einen UOW-Ereignismonitor ist aktiviert.

Dieser Parameter gibt an, ob Informationen zu einer UOW, die auch als Transaktion bezeichnet wird, an die aktiven UOW-Ereignismonitore gesendet werden sollen, wenn die UOW beendet wurde.

Wenn dieser Parameter inaktiviert wird, werden alle untergeordneten Parameter ebenfalls inaktiviert, jedoch ändern sich ihre Einstellungen, die in der Datenbankkonfigurationsdatei aufgezeichnet sind, nicht. Wenn dieser übergeordnete Parameter aktiviert wird, werden die aufgezeichneten Werte für die untergeordneten Parameter wirksam. Auf diese Weise kann die Datenerfassung global aktiviert bzw. inaktiviert werden.

Wenn der Parameter auf den Wert BASE gesetzt wird, werden Informationen zu allen UOWs, die auf dem Datenserver ausgeführt werden, an die aktiven UOW-Ereignismonitore gesendet, wenn die UOWs abgeschlossen wurden. Wenn der Parameter auf den Wert NONE gesetzt wird, werden Informationen an die UOW-Ereignismonitore nur zu den UOWs gesendet, die unter einer DB2-Workload ausgeführt werden, deren Klausel COLLECT UNIT OF WORK DATA auf den Wert BASE gesetzt ist. Die Standardeinstellung ist NONE.

Die Informationen, die am Ende einer UOW erfasst werden, enthalten die Anforderungsmesswerte auf Systemebene für diese UOW (z. B. die während der UOW genutzte CPU-Kapazität.). Die Erfassung dieser Anforderungsmesswerte wird unabhängig von der Erfassung der UOW-Daten gesteuert. Dies geschieht entweder mithilfe der Klausel COLLECT REQUEST METRICS in einer DB2-Servicesuperklasse oder indem der Datenbankkonfigurationsparameter **mon_req_metrics** auf den Wert BASE gesetzt wird. Wenn Sie die Erfassung von Anforderungsmessdaten nicht aktivieren, haben alle Anforderungsmesswerte, die als Teil der UOW-Daten erfasst werden, den Wert 0.

mon_uow_execlist - UOW-Ereignisüberwachung mit Liste ausführbarer Abschnitte (Konfigurationsparameter)

Dieser Parameter steuert die Generierung von UOW-Ereignissen mit eingeschlossenen Informationen zu Listen mit Kennungen der ausführbaren Abschnitte. Diese

Generierung erfolgt auf Datenbankebene für den UOW-Ereignismonitor. Der Datenbankkonfigurationsparameter **mon_uow_execlist** ist ein untergeordneter Parameter des Datenbankkonfigurationsparameters **mon_uow_data**.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Nächste UOW (Unit of Work)

Standardwert [Bereich]
OFF [OFF, ON]

Gültige Werte für diesen Parameter sind wie folgt:

OFF Die Erfassung der Liste der ausführbaren Abschnitte für einen UOW-Ereignismonitor ist inaktiviert.

ON Die Erfassung der Liste der ausführbaren Abschnitte für einen UOW-Ereignismonitor ist aktiviert.

mon_uow_pkglist - UOW-Ereignisüberwachung mit Paketliste (Konfigurationsparameter)

Dieser Parameter steuert die Generierung von UOW-Ereignissen (UOW, Unit of Work) mit eingeschlossenen Informationen zu Paketlisten. Diese Generierung erfolgt auf Datenbankebene für den UOW-Ereignismonitor. Der Datenbankkonfigurationsparameter **mon_uow_pkglist** ist ein untergeordneter Parameter des Datenbankkonfigurationsparameters **mon_uow_data**.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Nächste UOW (Unit of Work)

Standardwert [Bereich]
OFF [OFF, ON]

Gültige Werte für diesen Parameter sind wie folgt:

OFF Die Erfassung von Paketlisteninformationen für den UOW-Ereignismonitor ist inaktiviert.

ON Die Erfassung von Paketlisteninformationen für den UOW-Ereignismonitor ist aktiviert.

multipage_alloc - Zuordnung aus mehreren Seiten bestehender Datei aktiv

Die Zuordnung aus mehreren Seiten bestehender Dateien erhöht die Leistung beim Einfügen. Sie gilt nur für SMS-Tabellenbereiche. Wenn dieser Parameter aktiviert ist, sind alle SMS-Tabellenbereiche davon betroffen. Es ist keine Auswahl für einzelne SMS-Tabellenbereiche möglich.

Konfigurationstyp
Datenbank

Parametertyp
Informativ

Der Standardwert für den Parameter ist "Yes", d. h. die Zuordnung aus mehreren Seiten bestehender Dateien ist aktiviert.

Im Anschluss an eine Datenbankerstellung kann dieser Parameter nicht auf "No" gesetzt werden. Die Zuordnung mehrseitiger Dateien kann nach der Aktivierung nicht mehr inaktiviert werden. Das Tool **db2empfa** kann zur Aktivierung der Zuordnung mehrseitiger Dateien für eine Datenbank verwendet werden, für die dieses Merkmal inaktiviert ist.

newlogpath - Datenbankprotokollpfad ändern

Mit diesem Parameter können Sie eine Zeichenfolge mit bis zu 242 Byte angeben, um die Speicherposition der Protokolldateien zu ändern.

Konfigurationstyp
Datenbank

Parametertyp
Konfigurierbar

Standardwert [Bereich]
Null [gültiger Pfad oder gültige Einheit]

Die Zeichenfolge kann auf einen Pfadnamen oder auf eine Roheinheit verweisen.

Wichtig: Die Verwendung von Roheinheiten für die Datenbankprotokollierung ist seit Version 9.1 veraltet und wird in DB2 pureScale-Umgebungen nicht unterstützt. Sofern dies unterstützt wird, verwendet der Datenbankmanager die automatische, nicht gepufferte E/A für die aktiven Protokolldateien und gepufferte E/A für alle übrigen Protokolldateizugriffe.

Verweist die Zeichenfolge auf einen Pfadnamen, muss es sich um einen vollständig qualifizierten Pfad handeln und nicht um einen relativen Pfad.

Sowohl in DB2 pureScale-Umgebungen als auch in DB2 Enterprise Server Edition-Umgebungen werden die Datenbankpartitionsnummer und eine Protokolldatenstrom-ID automatisch an den Pfad angehängt. Beispiel: `/home/dbuser/dblogs/NODE0000/LOGSTREAM0000/`.

In einer DB2 pureScale-Umgebung wird das erste Member, das eine Verbindung zu der Datenbankprozesskonfiguration herstellt oder sie aktiviert, in diesen Protokollpfadparameter geändert. Der DB2-Datenbankmanager stellt sicher, dass der Pfad vorhanden ist und dass Lese- und Schreibzugriff für diesen Pfad besteht. Außerdem erstellt der Datenbankmanager memberspezifische Unterverzeichnisse für die Protokolldateien. Wenn eine dieser Operationen fehlschlägt, weist der DB2-Daten-

bankmanager den angegebenen Pfad zurück und schaltet die Datenbank mit dem alten Pfad online. Wenn der angegebene Pfad akzeptiert wird, wird der neue Wert auf jedes Member repliziert. Wenn ein Member beim Umschalten auf den neuen Pfad fehlschlägt, schlagen nachfolgende Versuche zum Aktivieren oder Verbinden dieses Pfads fehl (SQL5099N). Alle Member müssen denselben Protokollpfad verwenden.

Wenn Sie mit Replikation arbeiten wollen und Ihr Protokollpfad eine Roheinheit ist, muss der Konfigurationsparameter **overflowlogpath** konfiguriert werden.

Geben Sie zum Angeben einer Einheit eine Zeichenfolge an, die vom Betriebssystem als eine Einheit erkannt wird. Beispiel:

- Unter Windows: `\\.\d:` oder `\\.\PhysicalDisk5`

  Anmerkung: Damit Protokolle auf eine Einheit geschrieben werden können, muss Windows Version 4.0 mit Service Pack 3 oder höher installiert sein.
- Auf Linux- und UNIX-Plattformen: `/dev/rdblog8`

Anmerkung: Eine Einheit können Sie nur auf AIX-, Windows 2000-, Windows-, Solaris, HP-UX- und Linux-Plattformen angeben.

Diese Einstellung wird nur dann zum Wert des Parameters **logpath**, wenn die beiden folgenden Ereignisse eintreten:

- Die Datenbank ist in einem konsistenten Zustand, wie durch den Parameter **database_consistent** angegeben.
- Alle Anwendungen sind von der Datenbank getrennt.

Wenn die erste neue Verbindung zur Datenbank hergestellt wird, versetzt der Datenbankmanager die Protokolle an die neue, von **logpath** angegebene Speicherposition.

Im alten Protokollpfad befinden sich eventuell noch Protokolldateien. Diese Protokolldateien wurden eventuell nicht archiviert. Sie müssen sie möglicherweise manuell archivieren. Wenn Sie zudem für diese Datenbank eine Replikation ausführen, benötigt die Replikation eventuell weiterhin die vor der Protokollpfadänderung vorhandenen Protokolldateien. Wenn die Datenbank für die Verwendung der Protokollarchivierung konfiguriert ist und alle Protokolldateien entweder automatisch vom DB2-Datenbanksystem oder von Ihnen selbst manuell archiviert wurden, kann das DB2-Datenbanksystem die Protokolldateien zum Beenden des Replikationsprozesses abrufen. Andernfalls können Sie die Dateien aus dem alten Protokollpfad in den neuen Protokollpfad kopieren.

Wenn **logpath** oder **newlogpath** eine Roheinheit als Position zum Speichern der Protokolldateien angibt, ist die Protokollspiegelung, die durch **mirrorlogpath** angegeben wird, nicht zulässig. Wenn **logpath** oder **newlogpath** einen Dateipfad als Position zum Speichern von Protokolldateien angibt, ist die Protokollspiegelung zulässig, und **mirrorlogpath** muss ebenfalls einen Dateipfad angeben.

Empfehlung: Es empfiehlt sich, die Protokolldateien auf einer physischen Platte zu speichern, auf der **kein** großes Volumen an E/A-Operationen auftritt. Sie sollten beispielsweise die Protokolldateien nicht auf derselben Platte wie das Betriebssystem oder umfangreiche Datenbanken speichern. Dadurch werden effiziente Protokolliervorgänge ermöglicht und die zusätzliche Verarbeitungszeit wird reduziert, z. B. beim Warten auf E/A-Operationen.

Mit dem Datenbanksystemmonitor können Sie die Anzahl der E/A-Operationen für die Datenbankprotokollierung verfolgen.

Die Monitorelemente **log_reads** (Anzahl gelesener Protokollseiten) und **log_writes** (Anzahl geschriebener Protokollseiten) geben den auf die Datenbankprotokollierung bezogenen Umfang an E/A-Aktivität zurück. Sie können ein Tool zur Betriebssystemüberwachung verwenden, um Daten zu anderen Platten-E/A-Aktivitäten zu sammeln. Anschließend können Sie beide Arten von E/A-Aktivitäten vergleichen.

Verwenden Sie kein gemeinsam genutztes lokales Dateisystem oder Netzdateisystem als Protokollpfad für die Primärdatenbank und die Bereitschaftsdatenbank in einem DB2 High Availability Disaster Recovery-Datenbankpaar (HADR-Datenbankpaar). Die Primärdatenbank und die Bereitschaftsdatenbank haben jeweils Kopien der Transaktionsprotokolle, wobei die Primärdatenbank Protokolle an die Bereitschaftsdatenbank sendet. Wenn der Protokollpfad sowohl für die Primärdatenbank als auch für die Bereitschaftsdatenbank auf dieselbe physische Position zeigen würde, würden die Primärdatenbank und Bereitschaftsdatenbank dieselben physischen Dateien für die entsprechenden Kopien der Protokolle verwenden. Der Datenbankmanager gibt einen Fehler zurück, wenn er einen gemeinsam genutzten Protokollpfad erkennt.

num_db_backups - Anzahl der Datenbank-Backups

Dieser Parameter gibt die Anzahl der Datenbank-Backups an, die für eine Datenbank beibehalten werden sollen.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Transaktionsgrenzwert

Standardwert [Bereich]
12 [1 - 32 767]

Wird die angegebene Anzahl von Backups erreicht, werden alte Backups in der Datei des Recoveryprotokolls als abgelaufen markiert. Die Einträge in der Datei des Recoveryprotokolls für Tabellenbereichs-Backups und Ladekopie-Backups, die mit dem abgelaufenen Datenbank-Backup verbunden sind, werden ebenfalls als abgelaufen markiert. Wird ein Backup als abgelaufen markiert, können die physischen Backups von ihrem Speicherort (z. B. Platte, Band, TSM) entfernt werden. Das nächste Datenbank-Backup entfernt die abgelaufenen Einträge aus der Datei des Recoveryprotokolls.

Der Konfigurationsparameter **rec_his_retentn** sollte auf einen Wert gesetzt werden, der mit dem Wert von **num_db_backups** kompatibel ist. Wenn **num_db_backups** zum Beispiel auf einen hohen Wert gesetzt ist, muss der Wert für **rec_his_retentn** hoch genug sein, um die mit **num_db_backups** festgelegte Anzahl von Backups zu unterstützen.

num_freqvalues - Anzahl der häufigsten Werte

Mit diesem Parameter können Sie die Anzahl der „häufigsten Werte" angeben, die gesammelt werden, wenn die Option **WITH DISTRIBUTION** im Befehl **RUNSTATS** angegeben wird.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
`10 [0 - 32 767 ]`

Maßeinheit
Zähler

Wenn der Wert dieses Parameters erhöht wird, erhöht sich auch die Menge an Statistikzwischenspeicher (**stat_heap_sz**), die zum Sammeln statistischer Daten benötigt wird.

Die Statistik der „häufigsten Werte" unterstützt das Optimierungsprogramm beim Feststellen der Verteilung der Datenwerte innerhalb einer Spalte. Wenn der Wert erhöht wird, stehen dem Abfrageoptimierungsprogramm mehr Daten zur Verfügung. Allerdings ist zusätzlicher Katalogspeicher erforderlich. Wenn `0` angegeben wird, wird keine Statistik über häufige Werte erhoben, auch wenn Sie anfordern, dass statistische Informationen zur Datenverteilung gesammelt werden.

Sie können die Anzahl der häufigen Werte, die auf der Tabellen- oder Spaltenebene durch den Befehl **RUNSTATS** erfasst werden, auch mit dem Befehlsparameter **NUM_FREQVALUES** angeben. Wenn kein Wert angegeben wird, wird der Wert des Konfigurationsparameters **num_freqvalues** verwendet. Es ist einfacher, die Anzahl der häufigen Werte über den Befehl **RUNSTATS** zu ändern, als über den Datenbankkonfigurationsparameter **num_freqvalues**.

Durch Aktualisieren dieses Parameters kann das Optimierungsprogramm bessere Schätzwerte für die Selektivität einiger Vergleichselemente (=, <, >), die für nicht gleichmäßig verteilte Daten verwendet werden, erzielen. Exaktere Berechnungen der Selektivität können zur Auswahl effizienterer Zugriffspläne führen.

Nach der Änderung dieses Parameters müssen Sie wie folgt vorgehen:

- Führen Sie den Befehl **RUNSTATS** erneut aus, um Statistikdaten mit der geänderten Anzahl der Quantile zu sammeln.
- Führen Sie für alle Pakete mit statischen SQL- oder XQuery-Anweisungen einen Rebind durch.

Beim Verwenden des Befehls **RUNSTATS** haben Sie die Möglichkeit, die Anzahl der häufigen Werte zu begrenzen, die auf Tabellenebene und auf Spaltenebene gesammelt werden. Dadurch können Sie den in den Katalogen belegten Speicherbereich optimieren, indem Sie die Verteilungsstatistik für Spalten reduzieren, für die sie nicht genutzt werden kann, die Informationen für kritische Spalten jedoch weiterhin verwenden.

Empfehlung: Zur Aktualisierung dieses Parameters sollten Sie den Grad der Ungleichmäßigkeit der Daten in den wichtigsten Spalten (in den wichtigsten Tabellen) feststellen, für die in der Regel Auswahlvergleichselemente angegeben werden. Dies kann mithilfe einer SQL-Anweisung SELECT geschehen, die eine Rangfolge des Vorkommens jedes Werts in einer Spalte liefert. Dabei dürfen Sie gleichmäßig

verteilte, eindeutige, LONG- oder LOB-Spalten nicht berücksichtigen. Ein geeigneter praktischer Wert für diesen Parameter liegt im Bereich zwischen 10 und 100.

Beachten Sie, dass das Sammeln statistischer Daten über die häufigsten Werte erhebliche CPU- und Speicherressourcen (**stat_heap_sz**) erfordert.

num_iocleaners - Anzahl asynchroner Seitenlöschfunktionen

Mit diesem Parameter können Sie die Anzahl asynchroner Seitenlöschfunktionen für eine Datenbank angeben.

Konfigurationstyp
Datenbank

Parametertyp

- Konfigurierbar
- Nach Member in einer DB2 pureScale-Umgebung konfigurierbar

Standardwert [Bereich]
Automatic [0 – 255]

Anmerkung: Der Standardwert unterliegt nach der ursprünglichen Datenbankerstellung Änderungen durch den DB2 Configuration Advisor.

Maßeinheit
Zähler

Die Anzahl der Castout-Engines ist gleich dem Wert, der für den Konfigurationsparameter **num_iocleaners** festgelegt ist. Die maximale Anzahl der Castout-Engines beträgt jedoch 128.

Diese Seitenlöschfunktionen schreiben geänderte Seiten aus dem Pufferpool auf Platte, bevor der Bereich im Pufferpool von einem Datenbankagenten angefordert wird. Daher müssen Datenbankagenten in der Regel nicht die Auslagerung geänderter Seiten abwarten, bevor sie den Speicherbereich im Pufferpool nutzen können. Dadurch wird die Gesamtleistung der Datenbankanwendungen verbessert.

Die Seitenlöschfunktionen verringern außerdem die Zeit für Revocerys nach Systemausfällen, zum Beispiel aufgrund von Netzausfällen, weil der Inhalt der Datenbank auf der Platte zu einem gegebenen Zeitpunkt aktueller ist.

Wenn dieser Parameter auf AUTOMATIC gesetzt wird, richtet sich die Anzahl der gestarteten Seitenlöschfunktionen nach der Anzahl der auf dem aktuellen System konfigurierten physischen Systemeinheitskerne sowie nach der Anzahl der lokalen logischen Datenbankpartitionen in einer Umgebung mit partitionierten Datenbanken. Es wird in jedem Fall mindestens eine Seitenlöschfunktion gestartet, wenn dieser Parameter auf AUTOMATIC gesetzt ist.

Die Anzahl der zu startenden Seitenlöschfunktionen, wenn dieser Parameter auf AUTOMATIC gesetzt ist, wird anhand der folgenden Formel berechnet:

```
Anzahl Seitenlöschfunktionen = max( ceil( Anzahl CPUs / Anzahl lokaler
logischer DBpartitionen ) - 1, 1 )
```

Diese Formel stellt sicher, dass die Anzahl der Seitenlöschfunktionen nahezu gleichmäßig auf Ihre logischen Datenbankpartitionen verteilt wird und dass nicht mehr Seitenlöschfunktionen als physische Systemeinheitskerne vorhanden sind.

Anmerkung: Auf der HP-UX-Plattform wird in der Berechnung die Anzahl der logischen CPUs verwendet und nicht die Anzahl der physischen Systemeinheitskerne.

Empfehlung: Bei der Einstellung dieses Parameters müssen folgende Faktoren beachtet werden:

- Auslastung

  Umgebungen mit hohem Aufkommen an aktualisierenden Transaktionen können eventuell die Konfiguration weiterer Seitenlöschfunktionen erforderlich machen. Dies gilt nur, wenn **DB2_USE_ALTERNATE_PAGE_CLEANING** auf OFF gesetzt ist (dies ist die Standardeinstellung).
- Pufferpoolgrößen

  Umgebungen mit großen Pufferpools können eventuell auch die Konfiguration weiterer Seitenlöschfunktionen erforderlich machen. Dies gilt nur, wenn **DB2_USE_ALTERNATE_PAGE_CLEANING** auf OFF gesetzt ist (dies ist die Standardeinstellung).

Mithilfe des Datenbanksystemmonitors können Sie diesen Konfigurationsparameter optimieren, indem Sie die Informationen des Ereignismonitors zu Schreibaktivitäten aus einem Pufferpool heranziehen:

- Der Wert des Parameters kann verringert werden, wenn die beiden folgenden Bedingungen erfüllt sind:
  - **pool_data_writes** ist ungefähr gleich **pool_async_data_writes**.
  - **pool_index_writes** ist ungefähr gleich **pool_async_index_writes**.

  Anmerkung: Ein niedrigerer als der mit der Einstellung AUTOMATIC errechnete Wert ist nicht zu empfehlen.
- Der Wert des Parameters sollte erhöht werden, wenn eine der folgenden Bedingungen erfüllt ist:
  - **pool_data_writes** ist viel größer als **pool_async_data_writes**.
  - **pool_index_writes** ist viel größer als **pool_async_index_writes**.

num_ioservers - Anzahl von E/A-Servern

Mit diesem Parameter wird die Anzahl der E/A-Server für eine Datenbank definiert. Zu jedem beliebigen Zeitpunkt kann nur diese Anzahl von E/A-Servern zum Vorablesen und für Dienstprogramme für eine Datenbank aktiv sein.

Konfigurationstyp
Datenbank

Parametertyp
- Konfigurierbar
- Nach Member in einer DB2 pureScale-Umgebung konfigurierbar

Standardwert [Bereich]
Automatic [1 – 255]

Anmerkung: Der Standardwert unterliegt nach der ursprünglichen Datenbankerstellung Änderungen durch den DB2 Configuration Advisor.

Maßeinheit
Zähler

Zuordnung
Wenn eine Anwendung die Verbindung zu einer Datenbank herstellt

Freigabe
Wenn eine Anwendung die Verbindung zu einer Datenbank trennt

E/A-Server, die auch als Vorablesefunktionen bezeichnet werden, werden für Datenbankagenten verwendet, um Vorablese-E/A-Operationen und asynchrone E/A-Operationen durch Dienstprogramme wie BACKUP und RESTORE auszuführen. Ein E/A-Server wartet, während eine vom ihm eingeleitete E/A-Operation ausgeführt wird. Nicht vorabgelesene Ein-/Ausgaben werden direkt von den Datenbankagenten terminiert, sodass diese Ein-/Ausgaben nicht der Begrenzung durch den Parameter **num_ioservers** unterliegen.

Wenn dieser Parameter auf AUTOMATIC gesetzt ist, richtet sich die Anzahl der gestarteten Vorablesefunktionen nach den Einstellungen für die Parallelität der Tabellenbereiche in der aktuellen Datenbankpartition.

Wenn dieser Parameter auf AUTOMATIC gesetzt ist, wird die Anzahl der bei der Aktivierung der Datenbank zu startenden Vorablesefunktionen anhand der folgenden Formel berechnet:

```
Anzahl Vorablesefunktionen = max( max über alle Tabellenbereiche
( Parallelitätseinstellung ), 3 )
```

Die Einstellungen für die Parallelität werden mithilfe der Umgebungsvariablen **DB2_PARALLEL_IO** gesteuert.

Empfehlung: Um alle E/A-Einheiten des Systems voll auszunutzen, empfiehlt sich im Allgemeinen ein Wert, der um 1 oder 2 höher ist als die Anzahl der physischen Einheiten, auf denen sich die Datenbank befindet. Es ist besser, zusätzliche E/A-Server zu konfigurieren, da mit jedem E/A-Server nur eine gewisse zusätzliche Verarbeitungszeit erforderlich wird und alle nicht benötigten E/A-Server inaktiv bleiben.

num_log_span - Anzahl verwendeter Protokolldateien

Dieser Parameter gibt an, ob für die Menge an Protokolldateien, die eine Transaktion umfassen kann, ein Grenzwert definiert wurde und wie dieser lautet.

Konfigurationstyp
Datenbank

Parametertyp

- Online konfigurierbar
- Nach Member in einer DB2 pureScale-Umgebung konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
0 [0 - 65 535]

Maßeinheit
Zähler

Wenn der Wert nicht 0 ist, zeigt dieser Parameter die Anzahl aktiver Protokolldateien an, über die sich eine aktive Transaktion erstrecken kann.

Wird der Wert auf 0 gesetzt, besteht keine Begrenzung für die Anzahl der Protokolldateien, die eine einzelne Transaktion umfassen kann.

Verletzt eine Anwendung die Konfiguration von **num_log_span**, wird die Anwendung gezwungen, die Verbindung zur Datenbank zu beenden, und die Transaktion wird rückgängig gemacht.

Dieser Parameter kann zusammen mit dem Konfigurationsparameter **max_log** nützlich sein, wenn ein unbegrenzter aktiver Protokollspeicherbereich aktiviert wird. Wenn die Endlosprotokollierung aktiviert ist (d. h., für *logsecond* ist der Wert -1 definiert), werden Transaktionen nicht auf die Obergrenze der Anzahl von Protokolldateien (*logprimary* + *logsecond*) beschränkt. Wenn der Wert von *logprimary* erreicht wurde, beginnt DB2 mit der Archivierung der aktiven Protokolldateien, statt die Transaktion fehlschlagen zu lassen. Dies kann zu Problemen führen, wenn zum Beispiel eine Anwendung eine Transaktion mit langer Ausführungszeit enthält, die nicht festgeschrieben wurde. In einen solchen Fall wächst der Protokollspeicherbereich immer weiter an, was sich negativ auf die Leistung bei einer Recovery nach einem Systemabsturz auswirken könnte. Um dies zu verhindern, können Sie Werte für einen der Konfigurationsparameter **max_log** oder **num_log_span** (oder beide) angeben.

Anmerkung: Die folgenden DB2-Befehle sind von der Begrenzung ausgenommen, die vom Konfigurationsparameter **num_log_span** festgelegt wird: ARCHIVE LOG, BACKUP DATABASE, LOAD, REORG, RESTORE DATABASE und ROLLFORWARD DATABASE.

num_quantiles - Anzahl der Quantile für Spalten

Dieser Parameter steuert die Anzahl der Quantile, die gesammelt werden, wenn die Option **WITH DISTRIBUTION** im Befehl **RUNSTATS** angegeben wird.

Konfigurationstyp
Datenbank

Parametertyp

- Online konfigurierbar
- Nach Member in einer DB2 pureScale-Umgebung konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
`20 [0 - 32 767]`

Maßeinheit
Zähler

Wenn der Wert dieses Parameters erhöht wird, erhöht sich auch die Menge an Statistikzwischenspeicher (**stat_heap_sz**), die zum Sammeln statistischer Daten benötigt wird.

Die Statistik der „Quantile" unterstützt das Optimierungsprogramm beim Feststellen der Verteilung der Datenwerte innerhalb einer Spalte. Wenn der Wert erhöht wird, stehen dem Abfrageoptimierungsprogramm mehr Daten zur Verfügung; allerdings ist zusätzlicher Katalogspeicher erforderlich. Wenn der Wert `0` oder `1` angegeben wird, werden keine Quantil-Statistikdaten erhoben, auch wenn Sie anfordern, dass Verteilungsstatistikdaten gesammelt werden.

Sie können die Anzahl der Quantile, die auf der Tabellen- oder Spaltenebene durch den Befehl **RUNSTATS** erfasst werden, auch mit dem Befehlsparameter **NUM_QUANTILES** angeben. Wenn kein Wert angegeben wird, wird der Wert des Konfigurationspara-

meters **num_quantiles** verwendet. Es ist einfacher, die Anzahl der zu erfassenden Quantile über den Befehl **RUNSTATS** zu ändern, als über den Datenbankkonfigurationsparameter **num_quantiles**.

Durch Aktualisieren dieses Parameters können bessere Schätzwerte für die Auswahl von nicht gleichmäßig verteilten Daten mithilfe von Bereichsvergleichselementen erzielt werden. Unter anderem entscheidet das Optimierungsprogramm mithilfe dieser Informationen, ob eine Indexsuche oder eine Tabellensuche gewählt wird. (Beim Zugriff auf einen Bereich von Werten, die häufig vorkommen, ist eine Tabellensuche effizienter, während bei einem Bereich von Werten, die nicht häufig vorkommen, eine Indexsuche effizienter ist.)

Nach der Änderung dieses Parameters müssen Sie wie folgt vorgehen:

- Führen Sie den Befehl **RUNSTATS** erneut aus, um Statistikdaten mit der geänderten Anzahl der Quantile zu sammeln.
- Führen Sie für alle Pakete mit statischen SQL- oder XQuery-Anweisungen einen Rebind durch.

Beim Verwenden des Befehls **RUNSTATS** haben Sie die Möglichkeit, die Anzahl der erfassten Quantile sowohl auf Tabellenebene als auch auf Spaltenebene zu begrenzen. Dadurch können Sie den in den Katalogen belegten Speicherbereich optimieren, indem Sie die Verteilungsstatistik für Spalten reduzieren, in denen Sie nicht genutzt werden kann, und die Informationen jedoch weiter für kritische Spalten verwenden.

Empfehlung: Der Standardwert für diesen Parameter liefert in den meisten Fällen recht genaue Schätzwerte. Sie können in Betracht ziehen, den Wert zu erhöhen, wenn Sie bedeutende und konsistente Unterschiede zwischen folgenden Werten beobachten:

- Selektivitätsschätzwerte der EXPLAIN-Ausgabe und
- die aktuelle Selektivität von Bereichsvergleichselementen bei nicht gleichmäßig verteilte Spaltendaten.

Ein geeigneter praktischer Wert für diesen Parameter liegt im Bereich zwischen 10 und 50.

numarchretry - Anzahl der Wiederholungen bei Fehler

Mit diesem Parameter wird die Anzahl der Versuche angegeben, die das DB2-Datenbanksystem zum Archivieren einer Protokolldatei im primären oder sekundären Archivverzeichnis unternehmen soll, bevor es versucht, Protokolldateien im Verzeichnis für die Funktionsübernahme zu speichern.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp

- Online konfigurierbar

Standardwert [Bereich]
5 [0 - 65 535]

Dieser Parameter wird nur verwendet, wenn der Datenbankkonfigurationsparameter **failarchpath** definiert ist. Wenn **numarchretry** nicht definiert wird, setzt das DB2-Datenbanksystem die Versuche zur Archivierung im primären oder sekundären Protokollpfad kontinuierlich fort.

numsegs - Standardanzahl von SMS-Containern

Dieser Parameter ist in Version 9.5 veraltet, wird jedoch von Datenservern und Clients mit Versionen vor Version 9.5 weiterhin verwendet. Jeder für diesen Konfigurationsparameter angegebene Wert wird vom Datenbankmanager in DB2 Version 9.5 ignoriert.

Anmerkung: Die nachfolgenden Informationen gelten nur für Datenserver und Clients einer Version vor Version 9.5.

Konfigurationstyp
Datenbank

Parametertyp
Informativ

Maßeinheit
Zähler

Dieser Parameter gibt die Anzahl der Container an, die innerhalb der Standardtabellenbereiche erstellt werden. Er zeigt auch die Informationen an, die bei der Erstellung der Datenbank verwendet wurden, unabhängig davon, ob die Angabe im Befehl **CREATE DATABASE** explizit oder implizit war.

Dieser Parameter gilt nur für SMS-Tabellenbereiche. Er wird von der Anweisung CREATE TABLESPACE **nicht** verwendet.

number_compat - NUMBER-Kompatibilität (Datenbankkonfigurationsparameter)

Dieser Parameter gibt an, ob die Kompatibilitätssemantik, die mit dem Datentyp NUMBER verknüpft ist, auf die verbundene Datenbank angewendet wird.

Konfigurationstyp
Datenbank

Parametertyp
Informativ

Der Wert wird bei der Erstellung der Datenbank festgelegt und basiert auf der Einstellung der Registrierdatenbankvariablen **DB2_COMPATIBILITY_VECTOR** für die Unterstützung des Datentyps NUMBER. Der Wert kann nicht geändert werden.

overflowlogpath - Überlaufprotokollpfad

Der Parameter **overflowlogpath** gibt eine Speicherposition an, an der DB2-Datenbanken nach Protokolldateien suchen sollen, die für eine ROLLFORWARD-Operation benötigt werden; darüber hinaus gibt er eine Speicherposition für die aktiven Protokolldateien an, die aus dem Archiv abgerufen werden. Er gibt auch eine Speicherposition zum Suchen nach und Speichern von Protokolldateien an, die für die Verwendung der API db2ReadLog erforderlich sind.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
NULL [beliebiger gültiger Pfad]

Je nach Ihren Protokollierungsanforderungen kann dieser Parameter für verschiedene Funktionen verwendet werden.

- Mit dem Parameter **overflowlogpath** können Sie eine Speicherposition angeben, an der DB2-Datenbanken nach Protokolldateien suchen sollen, die für eine ROLLFORWARD-Operation benötigt werden. Dies bietet eine Alternative zur Verwendung der Option **OVERFLOW LOG PATH** mit dem **ROLLFORWARD**-Befehl. Anstatt jedes Mal die Option **OVERFLOW LOG PATH** mit dem **ROLLFORWARD**-Befehl anzugeben, können Sie einmal diesen Konfigurationsparameter festlegen. Wenn allerdings beides verwendet wird, überschreibt die Option **OVERFLOW LOG PATH** den Konfigurationsparameter **overflowlogpath** für diese bestimmte ROLLFORWARD-Konfiguration.
- Wenn **logsecond** auf **-1** gesetzt ist, können Sie über den Parameter **overflowlogpath** ein Verzeichnis angeben, in dem DB2 aus dem Archiv abgerufene aktive Protokolldateien speichert. Aktive Protokolldateien müssen für ROLLBACK-Operationen abgerufen werden, wenn sie sich nicht mehr im Pfad für aktive Protokolldateien befinden. Wird für **overflowlogpath** kein Wert angegeben, rufen die DB2-Datenbanken die Protokolldateien in den Pfad für aktive Protokolldateien ab. Mit dem Parameter **overflowlogpath** können Sie den DB2-Datenbanken zusätzliche Ressourcen zum Speichern von abgerufenen Protokolldateien bereitstellen. Dies bietet den Vorteil, den Ein-/Ausgabeaufwand auf verschiedene Datenträger zu verteilen und mehr Protokolldateien im Pfad für aktive Protokolldateien speichern zu können.
- Wenn Sie beispielsweise zur Replikation die API db2ReadLog benutzen müssen, geben Sie mit dem Parameter **overflowlogpath** eine Position an, an der die DB2-Datenbanken nach Protokolldateien suchen, die für diese API benötigt werden. Vor DB2 Version 8 wurde db2ReadLog als sqlurlog bezeichnet. Wenn die Protokolldatei nicht gefunden wird (weder im Pfad für aktive Protokolldateien noch im Überlaufprotokollpfad) und die Datenbank mit dem Parameter **logarchmeth1** oder **logarchmeth2** für die Protokollarchivierung konfiguriert ist, ruft DB2 die Protokolldatei ab. Mit dem Parameter **overflowlogpath** können Sie auch ein Verzeichnis angeben, in dem DB2-Datenbanken die abgerufenen Protokolldateien speichern. Dadurch wird der Ein-/Ausgabeaufwand für den Pfad für aktive Protokolldateien reduziert, und es können mehr Protokolldateien im Pfad für aktive Protokolldateien gespeichert werden.
- Wenn Sie für den Pfad für aktive Protokolldateien eine Roheinheit konfiguriert haben und den Parameter **logsecond** auf **-1** setzen oder die Anwendungsprogrammierschnittstelle db2ReadLog verwenden wollen, muss der Parameter **overflowlogpath** konfiguriert werden.
- Sie können eine Position für DB2-Datenbanken zum Abrufen von Protokolldateien angeben, die für eine Operation vom Typ **BACKUP DATABASE INCLUDE LOGS** erforderlich sind.

Zum Definieren von **overflowlogpath** müssen Sie eine Zeichenfolge mit maximal 242 Byte angeben. Diese Zeichenfolge muss auf einen Pfadnamen verweisen, der ein vollständig qualifizierter Pfadname und kein relativer Pfadname ist. Der Pfadname muss ein Verzeichnis sein, keine Roheinheit.

Anmerkung: Sowohl in DB2 pureScale-Umgebungen als auch in DB2 Enterprise Server Edition-Umgebungen werden die Datenbankpartitionsnummer und eine Protokolldatenstrom-ID automatisch an den Pfad angehängt. Beispiel: `/home/dbuser/dblogs/NODE0000/LOGSTREAM0000/`.

pagesize - Standardseitengröße für die Datenbank

Dieser Parameter enthält den Wert, der als Standardseitengröße bei der Erstellung der Datenbank verwendet wurde. Gültige Werte: 4 096, 8 192, 16 384 und 32 768. Wenn in dieser Datenbank ein Pufferpool oder ein Tabellenbereich erstellt wird, wird dieselbe Standardseitengröße verwendet.

Konfigurationstyp
Datenbank

Parametertyp
Informativ

pckcachesz - Größe des Paketcache

Dieser Parameter wird aus dem gemeinsam benutzten Datenbankspeicher zugeordnet und für das Caching (Zwischenspeichern) von Abschnitten statischer und dynamischer SQL- und XQuery-Anweisungen in einer Datenbank verwendet.

Konfigurationstyp
Datenbank

Parametertyp

- Online konfigurierbar
- Nach Member in einer DB2 pureScale-Umgebung konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]

32-Bit-Betriebssysteme
`Automatic [-1, 32 - 128 000]`

64-Bit-Betriebssysteme
`Automatic [-1, 32 - 2 147 483 646]`

Anmerkung: Der Standardwert unterliegt nach der ursprünglichen Datenbankerstellung Änderungen durch den DB2 Configuration Advisor.

Maßeinheit
Seiten (4 KB)

Zuordnung
Wenn die Datenbank initialisiert wird

Freigabe
Wenn die Datenbank heruntergefahren wird

In einem partitionierten Datenbanksystem gibt es für jede Datenbankpartition einen Paketcache.

Das Caching von Paketen ermöglicht dem Datenbankmanager, die interne Verarbeitungszeit zu verringern, da beim erneuten Laden eines Pakets kein Zugriff auf die Systemkataloge bzw. bei dynamischen SQL- oder XQuery-Anweisungen keine Kompilierung mehr erforderlich ist. Die Abschnitte werden im Paketcache behalten, bis eines der folgenden Ereignisse eintritt:

- Die Datenbank wird heruntergefahren.
- Das Paket oder die dynamische SQL- oder XQuery-Anweisung wird ungültig gemacht.
- Im Cache ist nicht mehr genügend Platz.

Dieses Caching des Abschnitts für eine statische oder dynamische SQL- oder XQuery-Anweisung kann die Leistung besonders dann verbessern, wenn dieselbe Anweisung mehrere Male von Anwendungen verwendet wird, die mit einer Datenbank verbunden sind. Dies ist insbesondere für eine Umgebung wichtig, die Transaktionen verarbeitet.

Der Wert AUTOMATIC für diesen Parameter aktiviert ihn für die automatische Leistungsoptimierung. Wenn der Parameter **self_tuning_mem** auf ON gesetzt ist, passt die Speicheroptimierungsfunktion die durch den Parameter **pckcachesz** gesteuerte Größe des Hauptspeicherbereichs an geänderte Auslastungsanforderungen nach Bedarf dynamisch an. Da die Speicheroptimierungsfunktion Speicherressourcen auf verschiedene Speicherverbraucher verteilt, müssen mindestens zwei Speicherverbraucher für die automatische Optimierung aktiviert sein, damit eine automatische Optimierung erfolgen kann.

Die automatische Optimierung dieses Konfigurationsparameters findet nur statt, wenn der Speicher mit automatischer Leistungsoptimierung für die Datenbank aktiviert ist (der Konfigurationsparameter **self_tuning_mem** muss auf ON gesetzt sein).

Wenn dieser Parameter auf den Wert -1 gesetzt ist, wird als Wert für die Berechnung der Seitenzuordnung das Achtfache des für den Konfigurationsparameter **maxappls** angegebenen Werts verwendet. Dies gilt jedoch nicht, wenn das Achtfache von **maxappls** kleiner als 32 ist. In diesem Fall setzt der Standardwert -1 den Parameter **pckcachesz** auf den Wert 32.

Empfehlung: Beim Optimieren dieses Parameters sollten Sie überlegen, ob der zusätzliche Speicher, der für den Paketcache reserviert wird, nicht günstiger für einen anderen Zweck zugeordnet werden könnte, z. B. für den Pufferpool oder einen Katalogcache. Aus diesem Grund sollten Sie zur Optimierung dieses Parameters Vergleichstests (Benchmark-Tests) durchführen.

Die optimale Einstellung dieses Parameters ist von besonderer Bedeutung, wenn zu Beginn mehrere Abschnitte verwendet werden und nur wenige Abschnitte wiederholt ausgeführt werden. Wenn der Cache zu groß ist, wird Speicher zum Behalten von Kopien der Anfangsabschnitte verschwendet.

Mithilfe der folgenden Monitorelemente können Sie ermitteln, ob Sie den Wert dieses Konfigurationsparameters anpassen sollten:

- **pkg_cache_lookups** (Zugriffe auf Paketcache)
- **pkg_cache_inserts** (Einfügungen in Paketcache)
- **pkg_cache_size_top** (obere Paketcachegrenze)
- **pkg_cache_num_overflows** (Überläufe des Paketcache)

Anmerkung: Der Paketcache ist ein Arbeitscache. Deshalb kann dieser Parameter nicht auf den Wert 0 gesetzt werden. Diesem Cache muss ausreichend Speicherplatz für alle Abschnitte der SQL- oder XQuery-Anweisungen zugeordnet sein, die momentan ausgeführt werden. Wenn mehr als der momentan benötigte Speicherplatz zugeordnet ist, werden Abschnitte zwischengespeichert. Diese Abschnitte

können einfach ausgeführt werden, wenn sie das nächste Mal benötigt werden, und müssen nicht erneut geladen oder kompiliert werden.

Der durch den Parameter **pckcachesz** angegebene Grenzwert ist ein veränderlicher Grenzwert. Dieser Grenzwert kann, falls erforderlich, überschritten werden, wenn in dem von den Datenbanken gemeinsam benutzten Speicher noch Speicherplatz verfügbar ist. Sie können mit dem Monitorelement **pkg_cache_size_top** den Höchstwert ermitteln, bis zu dem der Paketcache angewachsen ist. Mit dem Monitorelement **pkg_cache_num_overflows** können Sie ermitteln, wie häufig der durch den Parameter **pckcachesz** angegebene Grenzwert überschritten wurde.

priv_mem_thresh - Schwellenwert für privaten Speicher

Dieser Parameter ist in Version 9.5 veraltet, wird jedoch von Datenservern und Clients mit Versionen vor Version 9.5 weiterhin verwendet. Jeder für diesen Konfigurationsparameter angegebene Wert wird vom Datenbankmanager in DB2 Version 9.5 ignoriert.

Anmerkung: Die nachfolgenden Informationen gelten nur für Datenserver und Clients einer Version vor Version 9.5.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
`20 000 [-1; 32 - 112 000]`

Maßeinheit
Seiten (4 KB)

Dieser Parameter wird zur Festlegung des nicht benutzten privaten Agentenspeichers verwendet, der zugeordnet bleibt und zur Verwendung durch neue Agenten, die gestartet werden, zur Verfügung steht. Er gilt nicht für Linux- und UNIX-Plattformen.

Der Wert **-1** bewirkt, dass dieser Parameter den Wert des Parameters **min_priv_mem** verwendet.

Empfehlung: Bei der Einstellung dieses Parameters sollten die Abläufe, wann und wie Clients die Verbindung herstellen bzw. trennen, sowie der Speicherbedarf anderer Prozesse auf derselben Maschine berücksichtigt werden.

Wenn es nur eine kurze Phase gibt, während der viele Clients gleichzeitig mit der Datenbank verbunden sind, verhindert ein hoher Schwellenwert, dass ungenutzter Speicher freigegeben und für andere Prozesse verfügbar gemacht wird. Dies führt zu einer schlechten Speicherverwaltung, die andere Prozesse, für die Speicher erforderlich ist, beeinträchtigen kann.

Wenn die Anzahl gleichzeitig zugreifender Clients eher gleichmäßig hoch ist, aber zahlreiche Änderungen an dieser Zahl auftreten, stellt ein hoher Schwellenwert si-

cher, dass Speicher für die Clientprozesse verfügbar ist, und kann so die Verarbeitungszeit verringern, die für das Zuordnen und Freigeben von Speicher erforderlich ist.

rec_his_retentn - Aufbewahrungszeitraum für Recoveryprotokoll

Mit diesem Parameter wird angegeben, wie viele Tage die Protokolldaten zu Backups aufbewahrt werden.

Konfigurationstyp
Datenbank

Parametertyp
Konfigurierbar

Standardwert [Bereich]
366 [-1; 0 - 30 000]

Maßeinheit
Tage

Wenn die Datei des Recoveryprotokolls nicht benötigt wird, um Backups, Restores und Ladevorgänge festzuhalten, kann dieser Parameter auf einen kleineren Wert gesetzt werden.

Wenn **rec_his_retentn** auf -1 und **auto_del_rec_obj** auf OFF gesetzt ist, entspricht die Anzahl der Einträge, die Datenbankgesamtbackups anzeigen (sowie alle Tabellenbereichsbackups, die dem Datenbankbackup zugeordnet sind), dem Wert, der durch den Datenbankkonfigurationsparameter **num_db_backups** angegeben wird. Weitere Einträge in der Datei des Recoveryprotokolls können nur explizit mithilfe der verfügbaren Befehle oder APIs entfernt werden. Wenn **rec_his_retentn** auf -1 und **auto_del_rec_obj** auf ON gesetzt ist, wird die Protokolldatei nicht automatisch bereinigt und es werden keine Recovery-Objekte gelöscht.

Wenn **rec_his_retentn** auf 0 und **auto_del_rec_obj** auf OFF gesetzt ist, werden alle Einträge in der Protokolldatei - außer dem letzten Gesamtbackup - bereinigt. Wenn **auto_del_rec_obj** auf ON eingestellt ist, wird die automatische Bereinigung der Protokolldatei und das Löschen des Recovery-Objekts auf der Basis der Zeitmarke des Backups ausgeführt, der vom Datenbankkonfigurationsparameter **num_db_backups** ausgewählt wurde.

Unabhängig davon, wie kurz der Aufbewahrungszeitraum ist, werden das aktuelle Datenbankgesamtbackup und die zugehörige Restoregruppe immer zurückbehalten, sofern Sie nicht den Befehl **PRUNE HISTORY** mit der Option **FORCE** verwenden.

restore_pending - Restore anstehend

Dieser Parameter gibt an, ob sich die Datenbank im Status Restore anstehend befindet.

Konfigurationstyp
Datenbank

Parametertyp
Informativ

restrict_access - Eingeschränkter Datenbankzugriff (Konfigurationsparameter)

Dieser Parameter gibt an, ob die Datenbank mit dem eingeschränkten Satz an Standardaktionen erstellt wurde. Das heißt, ob sie mit der Klausel **RESTRICTIVE** im Befehl **CREATE DATABASE** erstellt wurde.

Konfigurationstyp
Datenbank

Parametertyp
Informativ

YES: Die Klausel **RESTRICTIVE** wurde bei der Erstellung dieser Datenbank im Befehl **CREATE DATABASE** verwendet.

NO: Die Klausel **RESTRICTIVE** wurde bei der Erstellung dieser Datenbank im Befehl **CREATE DATABASE** nicht verwendet.

rollfwd_pending - Aktualisierende Recovery anstehend

Über diesen Parameter werden Sie darüber informiert, ob eine aktualisierende Recovery erforderlich ist und wo sie erforderlich ist.

Konfigurationstyp
Datenbank

Parametertyp
Informativ

Dieser Parameter kann einen der folgenden Status anzeigen:

- DATABASE, d. h. eine aktualisierende Recovery ist für diese Datenbank erforderlich
- TABLESPACE, d. h. für mindestens einen Tabellenbereich muss eine aktualisierende Recovery durchgeführt werden
- NO, d. h. die Datenbank ist verwendbar und keine aktualisierende Recovery erforderlich

Die Recovery (mit **ROLLFORWARD DATABASE**) muss erfolgreich abgeschlossen sein, bevor auf die Datenbank bzw. den Tabellenbereich zugegriffen werden kann.

section_actuals - Ist-Daten für Abschnitt (Konfigurationsparameter)

Dieser Parameter aktiviert die Erfassung von Abschnitts-Ist-Daten (Laufzeitstatistikdaten, die während der Abschnittsausführung gemessen werden), sodass die Statistikdaten angezeigt werden können, wenn nachfolgend ein Ereignismonitor erstellt wird.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
UOW-Grenze

Standardwert [Bereich]
NONE [NONE, BASE]

Folgende Werte sind für diesen Parameter gültig:

NONE Gibt an, dass die Erfassung von Abschnitts-Ist-Daten inaktiviert ist.

BASE Gibt an, dass die Messung aktiviert ist und folgende Statistikdaten erfasst werden:

- Basiszähler für die Operatorkardinalität
- Statistikdaten für jedes beim Ausführen des Abschnitts referenzierte Objekt (nur DML-Anweisungen)

Nach dem Aktivieren der Abschnitts-Ist-Daten können Sie diese Daten mithilfe eines Aktivitätsereignismonitors erfassen und über eine mit der Prozedur EXPLAIN_FROM_ACTIVITY ausgeführte EXPLAIN-Operation für Abschnitte anzeigen.

Sie können diesen Parameter nicht aktivieren, wenn der Datenbankkonfigurationsparameter **auto_stats_prof** aktiviert ist (SQLCODE -5153).

self_tuning_mem - Speicher mit automatischer Leistungsoptimierung

Dieser Parameter legt fest, ob die Speicheroptimierungsfunktion verfügbare Speicherressourcen bei Bedarf unter den Speicherkonsumenten, für die die automatische Optimierung aktiviert ist, dynamisch verteilt.

Konfigurationstyp
Datenbank

Parametertyp

- Online konfigurierbar
- Nach Member in einer DB2 pureScale-Umgebung konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]

Umgebungen mit einer Einzeldatenbankpartition
ON [ON; OFF]

Umgebungen mit mehreren Datenbankpartitionen
OFF [ON; OFF]

In einer Datenbank, für die ein Upgrade von einer früheren Version durchgeführt wurde, behält der Parameter **self_tuning_mem** den alten Wert bei.

Anmerkung: Der Standardwert unterliegt nach der ursprünglichen Datenbankerstellung Änderungen durch den DB2 Configuration Advisor.

Da der Speicher zwischen Speicherkonsumenten aufgeteilt wird, müssen mindestens zwei Speicherkonsumenten für die automatische Optimierung aktiviert sein, damit die Speicheroptimierungsfunktion aktiviert werden kann. Wenn der Parameter **self_tuning_mem** auf ON gesetzt ist, jedoch weniger als zwei Speicherkonsumenten für die automatische Optimierung aktiviert sind, bleibt die Speicheroptimierungsfunktion inaktiv. (Eine Ausnahme bildet der Speicherbereich des

Sortierzwischenspeichers, der unabhängig davon optimiert werden kann, ob andere Speicherkonsumenten für die automatische Optimierung aktiviert sind oder nicht.)

Dieser Parameter hat in Umgebungen mit einer Einzeldatenbankpartition standardmäßig den Wert ON. In Umgebungen mit mehreren Datenbankpartitionen ist er standardmäßig auf OFF gesetzt.

Die folgenden Speicherkonsumenten können für die automatische Optimierung aktiviert werden:

- Paketcache (Steuerung durch den Konfigurationsparameter **pckcachesz**)
- Sperrenliste (Steuerung durch die Konfigurationsparameter **locklist** und **maxlocks**)
- Sortierspeicher (Steuerung durch die Konfigurationsparameter **sheapthres_shr** und **sortheap**)
- Gemeinsam genutzter Datenbankspeicher (Steuerung durch den Konfigurationsparameter **database_memory**)
- Pufferpools (Steuerung durch den Parameter SIZE der Anweisungen ALTER BUFFERPOOL und CREATE BUFFERPOOL)

  Anmerkung: In DB2 pureScale-Umgebungen können mit der Speicherfunktion mit automatischer Leistungsoptimierung nur die lokalen Pufferpools (LBPs) verwaltet werden. Der Speicher für Gruppenpufferpools (GBPs) wird mit dem Konfigurationsparameter **cf_gbp_sz** zugeordnet und gesteuert.

Zum Anzeigen der aktuellen Einstellung dieses Parameters verwenden Sie den Befehl **GET DATABASE CONFIGURATION** mit Angabe des Parameters **SHOW DETAIL**. Die folgenden Einstellungen, die für diesen Parameter zurückgegeben werden, sind möglich:

```
Speicher mit automatischer Leistungsoptimierung      (SELF_TUNING_MEM) = OFF
Speicher mit automatischer Leistungsoptimierung      (SELF_TUNING_MEM) = ON (Aktiv)
Speicher mit automatischer Leistungsoptimierung      (SELF_TUNING_MEM) = ON (Inaktiv)
Speicher mit automatischer Leistungsoptimierung      (SELF_TUNING_MEM) = ON
```

Diese Werte geben Folgendes an:

- ON (Aktiv): Mit der Speicheroptimierung wird der Speicher im System aktiv optimiert.
- ON (Inaktiv): Der Parameter ist zwar auf ON gesetzt, aber die Speicheroptimierung wird nicht ausgeführt, da weniger als zwei Speicherkonsumenten für die automatische Optimierung aktiviert sind oder der Quiescemodus der Datenbank aktiviert ist.
- ON (weder (Aktiv) noch (Inaktiv)): Aus einer Abfrage ohne die Option **SHOW DETAIL** bzw. ohne Datenbankverbindung.

In partitionierten Umgebungen zeigt der Konfigurationsparameter **self_tuning_mem** den Wert ON (Aktiv) nur für die Datenbankpartition, in der die Optimierungsfunktion ausgeführt wird. Auf allen anderen Knoten hat **self_tuning_mem** den Wert ON (Inaktiv). Wenn Sie feststellen möchten, ob die Speicheroptimierung in einer partitionierten Datenbank aktiv ist, müssen daher Sie den Parameter **self_tuning_mem** in allen Datenbankpartitionen überprüfen.

Wenn Sie ein Upgrade von einer früheren Version von DB2 auf DB2 Version 9 durchgeführt haben und planen, die Funktion für die automatische Speicheropti-

mierung zu nutzen, sollten Sie die folgenden Diagnoseanzeiger konfigurieren, um die Überprüfung von Schwellenwerten oder Statusangaben zu inaktivieren:

- Auslastung des gemeinsamen Sortierspeichers - **db.sort_shrmem_util**
- Prozentsatz der Sortierüberläufe - **db.spilled_sorts**
- Langfristige Auslastung des gemeinsamen Sortierspeichers - **db.max_sort_shrmem_util**
- Auslastung der Sperrenliste - **db.locklist_util**
- Rate für Sperreneskalation - **db.lock_escal_rate**
- Trefferquote für Paketcache - **db.pkgcache_hitratio**

Eines der Ziele der automatischen Speicheroptimierungsfunktion besteht darin, die Zuordnung von Speicher für einen Speicherkonsumenten zu vermeiden, wenn dies nicht unmittelbar erforderlich ist. Daher kann die Auslastung des Speichers, der einem Speicherkonsumenten zugeordnet ist, dem Wert 100 % sehr nahe kommen, bevor weiterer Speicher zugeordnet wird. Durch die Inaktivierung dieser Diagnoseanzeiger vermeiden Sie unnötige Alerts, die durch die hohe Speicherauslastung für einen Speicherkonsumenten ausgelöst werden.

Bei Instanzen, die in DB2 Version 9 erstellt werden, sind diese Diagnoseanzeiger standardmäßig inaktiviert.

seqdetect - Markierung für Sequenzerkennung und Vorauslesen (

Dieser Parameter steuert, ob der Datenbankmanager bei der E/A-Aktivität Sequenzerkennung oder Vorabzugriff durch Vorauslesen ausführen darf.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
Yes [Yes; No]

Der Datenbankmanager kann bei der Indexsuche E/A-Operationen überwachen und den E/A-Vorablesezugriff aktivieren, wenn sequenzielles Lesen von Seiten auftritt. Diese Art des sequenziellen Vorablesens ist die *Sequenzerkennung*. Während der Laufzeit kann der Typ des Vorablesezugriffs vom sequenziellen Vorablesen in Vorabzugriff durch Vorauslesen geändert werden, wenn festgestellt wird, dass das sequenzielle Vorablesen nicht gut genug funktioniert.

Wenn dieser Parameter auf 'No' gesetzt wird, sind sowohl Sequenzerkennung als auch Vorabzugriff durch Vorauslesen für den Befehl **INSPECT** mit der Option **INDEXDATA**, für den Befehl **RUNSTATS** und für alle Indexsuchen während der Ausführung der Abfrage inaktiviert. Im Befehl **REORG** mit der Option **CLEANUP** für Indizes wird durch Setzen des Parameters **seqdetect** auf 'No' jedoch lediglich das sequenzielle Vorablesen für Indizes inaktiviert.

Empfehlung: In den meisten Fällen sollte der Standardwert für diesen Parameter verwendet werden. Inaktivieren Sie den Parameter **seqdetect** nur dann, wenn andere Optimierungsversuche bisher nicht zur Behebung schwerwiegender Leistungsprobleme bei Abfragen geführt haben.

sheapthres_shr - Schwellenwert für Sortierspeicher für gemeinsame Sortiervorgänge

Dieser Parameter gibt einen veränderlichen Grenzwert für die Gesamtmenge des gemeinsam genutzten Datenbankspeichers an, die gleichzeitig von Sortierspeicherkonsumenten verwendet werden kann.

Konfigurationstyp
Datenbank

Gilt für
OLAP-Funktionen

Parametertyp

- Online konfigurierbar
- Nach Member in einer DB2 pureScale-Umgebung konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]

32-Bit-Plattformen
`Automatic [250 - 524 288]`

64-Bit-Plattformen
`Automatic [250 - 2 147 483 647]`

Anmerkung: Der Standardwert unterliegt nach der ursprünglichen Datenbankerstellung Änderungen durch den DB2 Configuration Advisor.

Maßeinheit
Seiten (4 KB)

Neben Sortiervorgängen gibt es noch weitere Konsumenten von Sortierspeicher, wie zum Beispiel Hash-Joins, logische Verknüpfungen von Indizes über AND (Index ANDing), logische Verknüpfungen von Blockindizes über AND, Mischjoins und speicherinterne Tabellen. Wenn sich die Gesamtgröße des gemeinsam genutzten Speichers, der von Konsumenten des gemeinsam genutzten Sortierspeichers beansprucht wird, dem Grenzwert **sheapthres_shr** nähert, wird ein Speicherdrosselungsmechanismus aktiviert, sodass bei nachfolgenden Anforderungen von Konsumenten des gemeinsam genutzten Sortierspeichers unter Umständen weniger Speicher als angefordert zugeordnet wird. Jedoch wird in jedem Fall mehr als das Minimum zugeordnet, das zur Ausführung der Operation erforderlich ist. Wenn der Grenzwert **sheapthres_shr** überschritten wird, erhalten alle Anforderungen von gemeinsam genutztem Sortierspeicher durch Sortierspeicherkonsumenten nur die minimale Größe an Speicher, die erforderlich ist, um die Operation zu beenden. Wenn die Gesamtmenge des gemeinsam genutzten Speichers für aktive gemeinsame Sortiervorgänge diesen Grenzwert erreicht, können nachfolgende Sortiervorgänge fehlschlagen (SQL0955C).

Wenn der Datenbankkonfigurationsparameter **sheapthres** den Wert `0` besitzt, verwenden alle Sortierspeicherkonsumenten für die Datenbank den gemeinsam genutzten Datenbankspeicher mit **sheapthres_shr** anstelle des privaten Sortierspeichers.

Wenn **sheapthres_shr** auf `AUTOMATIC` gesetzt ist, ist dieser Parameter für die automatische Leistungsoptimierung aktiviert. Dadurch ist die Speicheroptimierungsfunktion in der Lage, die durch diesen Parameter gesteuerte Größe des Hauptspeicherbereichs an geänderte Auslastungsanforderungen je nach Bedarf

dynamisch anzupassen. Da die Speicheroptimierungsfunktion Speicherressourcen auf verschiedene Speicherkonsumenten verteilt, müssen mindestens zwei Speicherkonsumenten für die automatische Optimierung aktiviert sein, damit eine automatische Optimierung erfolgen kann. Speicherkonsumenten sind zum Beispiel **sheapthres_shr**, **pckcachesz**, Pufferpool (jeder Pufferpool zählt als einer), **locklist** und **database_memory**.

Die automatische Optimierung von **sheapthres_shr** ist nur zulässig, wenn der Konfigurationsparameter **sheapthres** des Datenbankmanagers auf den Wert 0 gesetzt ist.

Der Wert des Parameters **sortheap** wird zusammen mit dem Parameter **sheapthres_shr** optimiert, sodass eine Inaktivierung der automatischen Optimierung des Parameters **sortheap** automatisch auch eine Inaktivierung der automatischen Optimierung des Parameters **sheapthres_shr** bewirkt. Die Aktivierung der automatischen Optimierung für den Parameter **sheapthres_shr** führt automatisch zu einer Aktivierung der automatischen Optimierung des Parameters **sortheap**.

Die automatische Optimierung dieses Konfigurationsparameters findet nur statt, wenn der Speicher mit automatischer Leistungsoptimierung für die Datenbank aktiviert ist (der Konfigurationsparameter **self_tuning_mem** muss auf ON gesetzt sein).

Wenn der Wert dieses Parameters online aktualisiert wird, wird der neue Wert nur von neuen Anforderungen für gemeinsam genutzten Sortierspeicher verwendet, die nach der Aktualisierung erfolgen. Es wird empfohlen, vor einer Verringerung des Werts von **sheapthres_shr** den Wert von **sortheap** herabzusetzen und vor einer Erhöhung des Werts von **sortheap** den Wert von **sheapthres_shr** heraufzusetzen.

Wenn der Wert des Datenbankkonfigurationsparameters **sheapthres** größer als 0 ist, ist der Parameter **sheapthres_shr** nur in zwei Fällen von Bedeutung:

- Für Ladevorgänge (LOAD) in eine XML-Tabelle ist gemeinsamer Sortierspeicher erforderlich. In diesem Fall muss der Parameterwert für **sheapthres_shr** ungleich null sein, andernfalls wird als Fehler gemeldet, dass der gemeinsame Sortierspeicher für dieses Dienstprogramm nicht zugeordnet werden konnte.
- Wenn der Konfigurationsparameter **intra_parallel** des Datenbankmanagers auf yes gesetzt ist, da keine gemeinsamen Sortiervorgänge durchgeführt werden, wenn **intra_parallel** auf no gesetzt ist.
- Wenn der Konzentrator aktiviert ist (d. h. wenn **max_connections** größer als **max_coordagents** ist), da für Sortiervorgänge, die einen mit der Option WITH HOLD deklarierten Cursor verwenden, Speicher aus dem gemeinsam genutzten Speicher zugeordnet wird.

smtp_server - SMTP-Server

Mit diesem Parameter wird ein SMTP-Server (SMTP = Simple Mail Transfer Protocol) identifziert. Dieser SMPT-Server überträgt E-Mails, die vom integrierten Modul UTL_MAIL gesendet werden.

Der Parameter aktzeptiert auch eine durch Kommas begrenzte Liste mit SMTP-Servern. UTL_MAIL versucht, E-Mails über den ersten SMTP-Server in der Liste zu senden. Wenn dieser SMTP-Server nicht verfügbar ist, wird der nächste Server in der Liste verwendet. Wenn alle Server in der durch Kommas begrenzten Liste nicht erreichbar sind, wird ein Fehler zurückgegeben.

Konfigurationstyp
Datenbank

Gilt für
Datenbankserver mit lokalen und fernen Clients

Datenbankserver mit lokalen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
Null [durch Kommas begrenzte Liste gültiger TCP/IP-Hostname für SMTP-Server]

softmax - Recoverybereich und Intervall für bedingte Prüfpunkte

Mit diesem Parameter wird die Häufigkeit von bedingten Prüfpunkten und der Recoverybereich festgelegt, die bei einer Recovery nach einem Systemabsturz für Unterstützung sorgen.

Konfigurationstyp
Datenbank

Parametertyp
Konfigurierbar

Standardwert [Bereich]

DB2 pureScale-Umgebung
`100 [ 1 – 65 535 ]`

Außerhalb von DB2 pureScale-Umgebungen
100 [1 – 100 * **`logprimary`**]

Anmerkung: Der Standardwert unterliegt nach der ursprünglichen Datenbankerstellung Änderungen durch den DB2 Configuration Advisor.

Maßeinheit
Prozentsatz der Größe einer primären Protokolldatei

Dieser Parameter hat folgende Funktionen:

- Beeinflussen der Anzahl von Protokolldateien, die für die Recovery nach einem Systemabsturz (z. B. nach einem Stromausfall) erforderlich sind. Wenn beispielsweise der Standardwert `100` verwendet wird, versucht der Datenbankmanager, die Anzahl der wiederherzustellenden Protokolldateien auf 1 zu halten. Wenn Sie `300` als Wert für diesen Parameter angeben, versucht der Datenbankmanager, die Anzahl der wiederherzustellenden Protokolldateien auf 3 zu halten.

  Beim Steuern der Anzahl der Protokolldateien, die für die Recovery nach einem Systemabsturz erforderlich sind, verwendet der Datenbankmanager diesen Parameter zum Starten der Seitenlöschfunktionen. Dadurch wird sichergestellt, dass Seiten, die älter sind als das angegebene Recoveryfenster, bereits auf Platte geschrieben sind.
- Festlegen der Frequenz der bedingten Prüfpunkte. Dies ist der Prozess zum Schreiben von Informationen in die Protokollsteuerdatei. Anhand dieser Informationen wird der Startpunkt im Protokoll ermittelt, wenn ein Neustart der Datenbank erforderlich ist.

Zum Zeitpunkt einer Datenbankstörung, die zum Beispiel durch einen Stromausfall verursacht wird, kann für in der Datenbank ausgeführte Änderungen Folgendes gelten:

- Die Änderungen wurden nicht mit COMMIT festgeschrieben, jedoch wurden die Daten im Pufferpool aktualisiert.
- Die Änderungen wurden mit COMMIT festgeschrieben, jedoch noch nicht vom Pufferpool auf die Festplatte geschrieben.
- Die Änderungen wurden mit COMMIT festgeschrieben und vom Pufferpool auf die Festplatte geschrieben.

Wenn eine Datenbank erneut gestartet wird, werden die Protokolldateien verwendet, um eine Recovery der Datenbank nach dem Systemabsturz auszuführen, die sicherstellt, dass die Datenbank in einem konsistenten Zustand verbleibt (d. h., alle mit COMMIT festgeschriebenen Transaktionen werden in der Datenbank nachvollzogen, und keine der nicht festgeschriebenen Transaktionen werden in der Datenbank nachvollzogen).

Der Datenbankmanager verwendet in einer Protokollsteuerdatei gespeicherte Informationen, um festzustellen, welche Datensätze aus der Protokolldatei in der Datenbank nachvollzogen werden müssen. (Der Datenbankmanager verwaltet tatsächlich zwei Kopien der Protokollsteuerdatei, `SQLOGCTL.LFH.1` und `SQLOGCTL.LFH.2`, sodass bei einer Beschädigung der einen Kopie immer noch die andere Kopie verwendet werden kann.) Diese Protokollsteuerdateien werden in regelmäßigen Abständen auf die Festplatte geschrieben, und der Datenbankmanager kann abhängig von der Frequenz dieses Ereignisses Protokollsätze festgeschriebener Transaktionen oder Protokollsätze zu Änderungen, die bereits aus dem Pufferpool auf Platte geschrieben wurden, nachvollziehen. Diese Protokollsätze haben keine Auswirkung auf die Datenbank, das Nachvollziehen dieser Protokollsätze führt jedoch zu einer gewissen zusätzlichen Verarbeitungszeit während des Neustarts der Datenbank.

Die Protokollsteuerdateien werden immer dann auf die Festplatte geschrieben, wenn eine Protokolldatei voll ist, und außerdem bei bedingten Prüfpunkten. Sie können diesen Konfigurationsparameter dazu verwenden, die Häufigkeit zusätzlicher bedingter Prüfpunkte zu steuern.

Die Ablaufsteuerung für bedingte Prüfpunkte wird mithilfe der Differenz zwischen dem „aktuellen Stand" und dem „aufgezeichneten Stand" festgelegt. Diese Differenz wird als Prozentsatz vom Wert des Parameters **logfilsiz** angegeben. Der „aufgezeichnete Stand" wird anhand des ältesten gültigen Protokollsatzes ermittelt, der in den Protokollsteuerdateien auf der Festplatte angegeben wird, während der „aktuelle Stand" anhand der Protokollsteuerinformationen im Hauptspeicher ermittelt wird. (Der älteste gültige Protokollsatz ist der erste Protokollsatz, der bei einem Recoveryprozess gelesen würde.) Der bedingte Prüfpunkt wird ausgelöst, wenn der nach der folgenden Formel berechnete Wert größer oder gleich dem Wert dieses Parameters ist:

```
( (Bereich zw. aufgezeichnetem u. aktuellem Stand) / logfilsiz ) * 100
```

Empfehlung: Sie können den Wert dieses Parameters erhöhen oder verringern, je nachdem, ob Ihr akzeptables Recoveryfenster größer oder kleiner als eine Protokolldatei ist. Wenn Sie den Wert dieses Parameters herabsetzen, wird der Datenbankmanager veranlasst, die Seitenlöschfunktionen häufiger auszulösen und häufiger bedingte Prüfpunkte anzusetzen. Diese Maßnahmen können die Anzahl der Protokollsätze, die verarbeitet werden müssen, und die Anzahl der überschüssigen Protokollsätze, die während der Recovery verarbeitet werden, verringern.

Beachten Sie jedoch, dass mehr Trigger von Seitenlöschfunktionen und häufigere bedingte Prüfpunkte die Verarbeitungszeit erhöhen, die mit der Protokollierung der Datenbank verbunden ist, was sich negativ auf die Leistung des Datenbankmanagers auswirken kann. Daneben können auch folgende Umstände dazu führen, dass häufigere bedingte Prüfpunkte die für den Neustart einer Datenbank benötigte Zeit nicht verkürzen:

- Es werden sehr lange Transaktionen mit wenigen Commitpunkten ausgeführt.
- Der Pufferpool ist sehr groß, und die Seiten mit den festgeschriebenen Transaktionen werden nicht sehr oft auf die Platte zurückgeschrieben. (Beachten Sie, dass durch das Verwenden asynchroner Seitenlöschfunktionen solche Situationen vermieden werden können.)

In beiden Fällen ändern sich die Protokollsteuerdaten im Hauptspeicher nur selten, und es ist nur dann sinnvoll, die Protokollsteuerdaten auf die Festplatte zu schreiben, wenn sie sich geändert haben.

sortheap - Sortierspeichergröße

Dieser Parameter definiert die maximale Anzahl von Seiten des privaten bzw. des gemeinsamen Speichers für eine Operation, die Sortierspeicher erfordert.

Konfigurationstyp
Datenbank

Parametertyp

- Online konfigurierbar
- Nach Member in einer DB2 pureScale-Umgebung konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]

32-Bit-Plattformen
`Automatic [16 - 524 288]`

64-Bit-Plattformen
`Automatic [16 - 4 194 303]`

Anmerkung: Der Standardwert unterliegt nach der ursprünglichen Datenbankerstellung Änderungen durch den DB2 Configuration Advisor.

Maßeinheit
Seiten (4 KB)

Zuordnung
Wie zur Ausführung von Operationen erforderlich, die Sortierspeicher benötigen

Freigabe
Wenn Operationen, die Sortierspeicher benötigen, abgeschlossen sind

Wenn es sich um einen privaten Sortiervorgang handelt, bezieht sich dieser Parameter auf den privaten Agentenspeicher. Handelt es sich um einen gemeinsamen Sortiervorgang, bezieht sich dieser Parameter auf den gemeinsamen Datenbankspeicher. Jeder Sortiervorgang verwendet einen getrennten Sortierspeicher, der bei Bedarf vom Datenbankmanager zugeordnet wird. Dieser Sortierspeicher ist der Bereich, in dem Daten sortiert werden. Bei Steuerung durch das Optimierungspro-

gramm wird anhand der vom Optimierungsprogramm bereitgestellten Informationen ein kleinerer als der durch diesen Parameter angegebene Sortierspeicher zugeordnet.

Der Wert AUTOMATIC für diesen Parameter aktiviert ihn für die automatische Leistungsoptimierung. Dadurch ist die Speicheroptimierungsfunktion in der Lage, die durch diesen Parameter gesteuerte Größe des Hauptspeicherbereichs an geänderte Auslastungsanforderungen je nach Bedarf dynamisch anzupassen.

Der Wert des Parameters **sortheap** wird zusammen mit dem Parameter **sheapthres_shr** optimiert, sodass eine Inaktivierung der automatischen Optimierung des Parameters **sortheap** automatisch auch eine Inaktivierung der automatischen Optimierung des Parameters **sheapthres_shr** bewirkt. Die Aktivierung der automatischen Optimierung für den Parameter **sheapthres_shr** führt automatisch zu einer Aktivierung der automatischen Optimierung des Parameters **sortheap**. Der Parameter **sortheap** kann jedoch für die automatische Optimierung aktiviert werden, ohne dass der Parameter **sheapthres_shr** auf AUTOMATIC gesetzt ist.

Die automatische Optimierung von **sortheap** ist nur zulässig, wenn der Datenbankkonfigurationsparameter **sheapthres** auf den Wert 0 gesetzt ist.

Die automatische Optimierung dieses Konfigurationsparameters findet nur statt, wenn der Speicher mit automatischer Leistungsoptimierung für die Datenbank aktiviert ist (der Konfigurationsparameter **self_tuning_mem** muss auf ON gesetzt sein).

Empfehlung: Bei der Arbeit mit dem Sortierspeicher sind folgende Faktoren zu beachten:

- Geeignete Indizes können die Verwendung des Sortierspeichers minimieren.
- Die folgenden Operationen verwenden Sortierspeicher. Erhöhen Sie den Wert dieses Parameters, wenn eines der folgenden Verfahren verwendet wird:
  - Hash-Join-Puffer
  - Logisches Verknüpfen von Blockindizes über AND
  - Mischjoins
  - Tabellen im Speicher
  - Dynamische Bitzuordnungen (die für logisches Verknüpfen von Indizes über AND und Sternjoins verwendet werden)
  - Tabellenwarteschlangen (wenn eine Abfrage von mehreren Datenbankagenten verarbeitet wird)
  - Partielle frühe DISTINCT-Operationen
  - Partielle frühe Spaltenberechnungen
- Erhöhen Sie den Wert dieses Parameters, wenn häufig umfangreiche Sortiervorgänge ausgeführt werden müssen.
- Wenn Sie den Wert dieses Parameters erhöhen, sollten Sie überprüfen, ob auch die Werte der Parameter **sheapthres** und **sheapthres_shr** in der Konfigurationsdatei des Datenbankmanagers angepasst werden müssen.
- Die Größe des Zwischenspeichers für Sortierlisten wird vom Optimierungsprogramm zur Bestimmung der Zugriffspfade verwendet. Wenn Sie diesen Parameter geändert haben, sollten Sie für die Anwendungen eventuell einen Rebind durchführen (mit dem Befehl **REBIND**).

Nach einer Aktualisierung des Werts für **sortheap** beginnt der Datenbankmanager für sämtliche aktuellen und neuen Sortiervorgänge unverzüglich mit der Verwendung dieses neuen Werts.

sql_ccflags - Markierungen für bedingte Kompilierung

Dieser Parameter enthält eine Liste von Werten für bedingte Kompilierungen, die bei der bedingten Kompilierung ausgewählter SQL-Anweisungen verwendet werden.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Der Wert des Parameters **sql_ccflags** muss mindestens ein Paar aus Name und Wert enthalten, wobei der Name durch das Doppelpunktzeichen vom Wert getrennt ist. Jedes Name/Wert-Paar muss durch ein Komma vom vorangehenden Paar getrennt werden. Der Name muss eine gültige und normale SQL-Kennung sein. Der Wert muss eine boolesche SQL-Konstante (BOOLEAN), eine ganzzahlige SQL-Konstante (INTEGER) oder das Schlüsselwort NULL sein. Die maximale Länge der Zeichenfolge beträgt 1023 Byte.

Wenn der Wert des Parameters **sql_ccflags** aktualisiert wird, wird er nicht sofort auf Gültigkeit geprüft. Die Prüfung erfolgt, wenn der Wert zum ersten Mal zur Initialisierung des Sonderregisters CURRENT SQL_CCFLAGS verwendet wird; das heißt, wenn eine Verbindung zur Datenbank zum ersten Mal auf CURRENT SQL_CCFLAGS verweist oder eine Abfragedirektive in einer SQL-Anweisung angetroffen wird. Stellen Sie nach einer Aktualisierung des Werts des Parameters **sql_ccflags** eine Verbindung zur Datenbank her und fragen Sie das Sonderregister mithilfe der folgenden Anweisung ab: `VALUES CURRENT SQL_CCFLAGS`.

stat_heap_sz - Größe des Statistikzwischenspeichers

Mit diesem Parameter wird die *maximale* Größe des Zwischenspeichers angegeben, der bei Erfassung statistischer Daten mit dem Befehl **RUNSTATS** verwendet wird.

Die von diesem Parameter festgelegte Integritätsbedingung gilt für jede **RUNSTATS**-Operation.

Ab Version 9.5 hat dieser Datenbankkonfigurationsparameter den Standardwert `AUTOMATIC`, was bedeutet, dass er nach Bedarf erhöht wird, bis entweder der Grenzwert des Parameters **appl_memory** oder des Parameters **instance_memory** erreicht ist.

Konfigurationstyp
Datenbank

Parametertyp

- Online konfigurierbar
- Nach Member in einer DB2 pureScale-Umgebung konfigurierbar

Standardwert [Bereich]
`Automatic [1 096 - 524 288]`

Anmerkung: Der Standardwert unterliegt nach der ursprünglichen Datenbankerstellung Änderungen durch den DB2 Configuration Advisor.

Maßeinheit
Seiten (4 KB)

Zuordnung
Wenn das Dienstprogramm **RUNSTATS** gestartet wird

Freigabe
Wenn das Dienstprogramm **RUNSTATS** beendet ist

Empfehlung: Der Speicherbedarf für **RUNSTATS** hängt von mehreren Faktoren ab. Beim Einsatz von mehr Statistikoptionen, z. B. wenn LIKE-Statistiken oder DETAILED-Indexstatistiken erfasst werden, ist mehr Speicher erforderlich. Wenn Spaltenstatistiken erfasst werden, erhöht sich der Speicherbedarf durch die Erfassung von Statistiken für eine höhere Zahl Spalten. Wenn Verteilungsstatistiken erfasst werden, ist für die Erfassung einer höheren Zahl von häufigsten und/oder Quantilwerten mehr Speicher erforderlich. Es wird empfohlen, den Standardwert AUTOMATIC zu verwenden.

stmt_conc - Anweisungskonzentrator (Konfigurationsparameter)

Mit diesem Konfigurationsparameter wird das Standardverhalten des Anweisungskonzentrators festgelegt.

Konfigurationstyp
Datenbank

Parametertyp

- Online konfigurierbar
- Nach Member in einer DB2 pureScale-Umgebung konfigurierbar

Weitergabeklasse
Anweisungsgrenzwert

Standardwert [Bereich]
OFF [OFF, LITERALS]

Dieser Konfigurationsparameter aktiviert die Anweisungskonzentration für dynamische Anweisungen. Die Einstellung in der Datenbankkonfiguration wird nur verwendet, wenn der Client den Anweisungskonzentrator nicht explizit aktiviert oder inaktiviert.

Wenn aktiviert, ändert der Anweisungskonzentrator dynamische Anweisungen, um eine erweiterte gemeinsame Nutzung von Paketcacheeinträgen zu ermöglichen.

Der Anweisungskonzentrator wird inaktiviert, wenn der Konfigurationsparameter auf den Wert OFF gesetzt wird. Durch den Wert LITERALS wird der Anweisungskonzentrator aktiviert. Wenn der Anweisungskonzentrator aktiviert ist, können SQL-Anweisungen, die bis auf die Werte von Literalen identisch sind, Paketcacheeinträge gemeinsam nutzen.

Wenn der Parameter **stmt_conc** auf den Wert LITERALS gesetzt ist, verwenden die Anweisungen

```
SELECT FIRSTNME, LASTNAME FROM EMPLOYEE WHERE EMPNO='000020'
```

und

```
SELECT FIRSTNME, LASTNAME FROM EMPLOYEE WHERE EMPNO='000070'
```

denselben Eintrag im Paketcache gemeinsam. In diesem Fall verwendet der Eintrag im Paketcache die Anweisung

```
SELECT FIRSTNME, LASTNAME FROM EMPLOYEE WHERE EMPNO=:L0
```

und das DB2-Datenbanksystem stellt den Wert für

```
:L0 (entweder '000020' oder '000070')
```

auf der Basis des in den ursprünglichen Anweisungen verwendeten Literals bereit.

Dieser Parameter kann erhebliche Auswirkungen auf die Planauswahl haben, da er den Anweisungstext ändert. Der Anweisungskonzentrator darf nur verwendet werden, wenn ähnliche Anweisungen im Paketcache ähnliche Pläne haben. Wenn zum Beispiel unterschiedliche Literalwerte in einer Anweisung zu sehr unterschiedlichen Plänen führen, darf der Anweisungskonzentrator nicht auf den Wert LITERALS gesetzt werden.

Der Konfigurationsparameter **stmt_conc** ist möglicherweise die Ursache dafür, dass die Längenattribute für die Zeichenfolgeliterale VARCHAR und VARGRAPHIC größer sind als die Länge des Zeichenfolgeliterals.

stmtheap - Größe des Anweisungszwischenspeichers

Dieser Parameter legt die Größe des Anweisungszwischenspeichers fest, der als Arbeitsbereich für den SQL- oder XQuery-Compiler während der Kompilierung einer SQL- oder XQuery-Anweisung verwendet wird.

Konfigurationstyp
Datenbank

Parametertyp
Online konfigurierbar

Nach Member in einer DB2 pureScale-Umgebung konfigurierbar

Weitergabeklasse
Anweisungsgrenzwert

Standardwert [Bereich]

Für 32-Bit-Plattformen
AUTOMATIC [128 - 524288]

- Datenbankserver mit lokalen und fernen Clients: Die Standardeinstellung ist AUTOMATIC mit einem zugrunde liegenden Wert von 2048.
- Es ist auch möglich, ohne das Attribut AUTOMATIC für diesen Parameter nur einen festen Wert zu definieren.

Für 64-Bit-Plattformen
AUTOMATIC [128 - 524288]

- Datenbankserver mit lokalen und fernen Clients: Die Standardeinstellung ist AUTOMATIC mit einem zugrunde liegenden Wert von 8192.
- Es ist auch möglich, ohne das Attribut AUTOMATIC für diesen Parameter nur einen festen Wert zu definieren.

Anmerkung: Der Standardwert unterliegt nach der ursprünglichen Datenbankerstellung Änderungen durch den DB2 Configuration Advisor.

Maßeinheit
Seiten (4 KB)

Zuordnung
Für jede Anweisung während des Vorkompilierens oder des Bindens

Freigabe
Wenn das Vorkompilieren oder Binden der betreffenden Anweisung abgeschlossen ist

Dieser Bereich bleibt nicht permanent zugeordnet, sondern wird für die Verarbeitung jeder SQL- oder XQuery-Anweisung einzeln zugeordnet und anschließend wieder freigegeben. Beachten Sie, dass dieser Arbeitsbereich bei dynamischen SQL- oder XQuery-Anweisungen während der Ausführung Ihres Programms, bei statischen SQL- oder XQuery-Anweisungen hingegen während des Bindens, und nicht während der Ausführung des Programms, verwendet wird.

Für den Parameter **stmtheap** kann die Einstellung AUTOMATIC mit einem zugrunde liegenden Wert definiert werden oder ein fester Wert. Wird die Einstellung AUTOMATIC definiert, erzwingt der zugrunde liegende Wert eine Begrenzung der Speichermenge, die für eine einzelne Kompilierung mit dynamischer Joinaufzählung zugeordnet wird. Wird eine Speicherbegrenzung festgestellt, wird die Anweisungskompilierung mit einer schnellen Joinaufzählung (Greedy Join Enumeration) und einem uneingeschränkten Anweisungszwischenspeicher erneut gestartet. Die einzige Begrenzung besteht in der Menge des verbleibenden Anwendungsspeichers (**appl_memory**), Instanzspeichers (**instance_memory**) oder Systemspeichers. Wenn die schnelle Joinaufzählung erfolgreich abgeschlossen ist, wird einen Warnung SQL0437W (Ursachencode 1) an die Anwendung zurückgegeben. Stellt die schnelle Joinaufzählung ebenfalls eine Speicherbegrenzung fest, schlägt die Anweisungsvorbereitung mit dem Fehler SQL0101N fehl.

Beispiel: **db2 update db cfg for SAMPLE using STMTHEAP 8192 AUTOMATIC** führt zu einer Anweisungszwischenspeicherbegrenzung von 8192 * 4 KB (32 MB) für dynamische Joinaufzählungen und zu einem uneingeschränkten Anweisungszwischenspeicher für schnelle Joinaufzählungen.

Wenn für den Parameter **stmtheap** ein fester Wert definiert wird, gilt die Begrenzung sowohl für die dynamische als auch für die schnelle Joinaufzählung. Stellt die dynamische Joinaufzählung eine Speicherbegrenzung fest, wird eine schnelle Joinaufzählung versucht, wobei dieselbe feste Anweisungszwischenspeicherbegrenzung verwendet wird. Die möglicherweise zurückgegebenen Warnungen oder Fehler entsprechen denen für die Einstellung AUTOMATIC.

Beispiel: **db2 update db cfg for SAMPLE using STMTHEAP 8192** führt zu einer Anweisungszwischenspeicherbegrenzung von 8192 * 4 KB (32 MB) für dynamische und schnelle Joinaufzählungen.

Wenn die Laufzeitleistung Ihrer Abfrage nicht ausreicht, können Sie erwägen, den Wert des Konfigurationsparameters **stmtheap** (entweder den der Einstellung AUTOMATIC zugrunde liegenden Wert oder einen festen Wert) zu erhöhen, um sicherzustellen, dass die dynamisch programmierte Joinaufzählung (Dynamic Programming Join Enumeration) erfolgreich ist. Wenn Sie den Konfigurationsparameter **stmtheap** zum Optimieren der Leistung einer Abfrage aktualisieren, müssen Sie veranlassen, dass die Anweisung erneut kompiliert wird, damit das Abfrageoptimierungsprogramm einen neuen Zugriffsplan erstellen kann, um die geänderte Menge von Anweisungszwischenspeicher auszunutzen.

Anmerkung: Dynamisch programmierte Joinaufzählungen treten nur bei der Optimierungsklasse 3 und höher auf (5 ist der Standardwert).

suspend_io - Status der Datenbank-E/A-Operationen (Konfigurationsparameter)

Dieser Parameter gibt an, ob die E/A-Schreiboperationen für eine Datenbank ausgesetzt sind oder ausgesetzt werden.

Konfigurationstyp
Datenbank

Parametertyp
Informativ

Mögliche Werte sind `YES`, `IN_PROGRESS` und `NO`.

systime_period_adj - Temporalen Zeitraum für SYSTEM_TIME anpassen (Datenbankkonfigurationsparameter)

Dieser Datenbankkonfigurationsparameter gibt an, welche Aktion ausgeführt werden soll, wenn eine Protokollzeile für eine temporale Tabelle für Systemzeitraum mit einer Endzeitmarke generiert wird, die vor der Anfangszeitmarke liegt.

Dies kann vorkommen, wenn zwei verschiedene Transaktionen bei dem Versuch in Konflikt geraten, dieselbe Zeile in einer temporalen Tabelle für Systemzeitraum zu aktualisieren. Dies kann auch durch das Anpassen der Systemuhr verursacht werden (z. B. wenn die Systemuhr am Ende der Sommerzeit um eine Stunde zurück gestellt wird).

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
NO [NO, YES]

Wenn eine Zeile in einer temporalen Tabelle für Systemzeitraum aktualisiert wird, wird eine Protokollzeile mit einem Zeitraum für SYSTEM_TIME generiert, der angibt, in welchem Zeitraum die Daten in der Protokollzeile aktuell waren. Der Wert in der Beginnspalte für Zeile gibt an, wann die Daten in der Protokollzeile aktuell wurden. Der Wert in der Endspalte für Zeile gibt an, ab wann die Daten in der Protokollzeile nicht mehr aktuell waren.

Das nachfolgende Beispiel zeigt, wie zwei in Konflikt stehende Transaktionen möglicherweise eine Protokollzeile generieren, deren Endspalte für Zeile eine Zeitmarke enthält, die vor der Zeitmarke der Beginnspalte für Zeile liegt.

1. Die Transaktion TRA enthält einen Beginnspaltenwert, der für eine Zeile in einer temporalen Tabelle für Systemzeitraum generiert wurde, die an dem in der Zeitmarke T1 angegebenen Zeitpunkt von der Transaktion aktualisiert wird.

2. Die Transaktion TRB enthält einen Beginnzeilenwert, der für dieselbe Zeile generiert wurde, die an dem in der Zeitmarke T2 angegebenen Zeitpunkt aktualisiert wird (dabei gilt Folgendes: T1 < T2).
3. Die Transaktion TRB generiert eine Protokollzeile und schreibt sie mit Commit fest.
4. Die Transaktion TRA generiert die zugehörige Protokollzeile und schreibt sie mit Commit fest.

Nach dieser Abfolge von Ereignissen würde die Endzeitmarke der für Transaktion TRA generierten Protokollzeile kleiner als die Startzeitmarke sein.

Der Datenbankmanager kann sicherstellen, dass die Endzeitmarke der generierten Protokollzeilen stets größer ist als die Startzeitmarke, indem bei Auftreten eines Konflikts Zeitmarkenanpassungen vorgenommen werden oder für eine der beteiligten Transaktionen ein Rollback durchgeführt wird.

NO Wenn die Endzeitmarke einen kleineren Wert als die Startzeitmarke aufweist, werden für eine einzufügende Protokollzeile keine Anpassungen des Zeitmarkenwerts durchgeführt. Stattdessen schlägt die Transaktion fehl, die versucht hat, die Protokollzeile einzufügen, und ein Fehler wird zurückgegeben (SQLSTATE 57062, SQLCODE SQL20528N). Dadurch, dass Anpassungen nicht zugelassen sind, wird sichergestellt, dass alle während der Transaktion generierten Protokollzeilen dieselbe Endzeitmarke aufweisen und unter Verwendung dieser Endzeitmarke einfach ermittelt werden können.

YES Bei Zeitmarkenkonflikten werden der Zeitmarkenwert der Beginnspalte für Zeile in der temporalen Tabelle für Systemzeitraum und der Endzeitmarkenwert für die generierte Protokollzeile angepasst. Die Anpassung besteht aus der Änderung der Endzeitmarke in einen Wert, der um 1 Mikrosekunde größer als der Wert der Startzeitmarke ist. Dadurch wird sichergestellt, dass die Endzeitmarke größer ist als die Startzeitmarke für die Protokollzeile.Es wird eine Nachricht zurückgegeben, die angibt, dass eine Anpassung ausgeführt wurde (SQLSTATE 01695, SQLCODE SQL5191W).

Falls keine Zeitmarkenanpassungen erforderlich sind, wird SQLCODE DB20000I zurückgegeben.

Anwendungsprogrammierer sollten die Verwendung von SQLCODE- oder SQLSTATE-Werten erwägen, um diese auf die Anpassung von Zeitmarkenwerten bezogenen Rückkehrcodes über SQL-Anweisungen zu handhaben.

territory - Datenbankgebiet

Dieser Parameter zeigt das Gebiet an, mit dem die Datenbank erstellt wurde. Der Parameter **territory** wird vom Datenbankmanager bei der Verarbeitung von Daten verwendet, für die das Gebiet von Bedeutung ist.

Konfigurationstyp
Datenbank

Parametertyp
Informativ

trackmod - Geänderte Seiten protokollieren

Mit diesem Parameter wird angegeben, ob der Datenbankmanager Datenbankänderungen verfolgt, damit das Backup-Dienstprogramm feststellen kann, welche Un-

tergruppen der Datenbankseiten bei einem inkrementellen Backup zu berücksichtigen und eventuell in das Backup-Image aufzunehmen sind.

Konfigurationstyp
Datenbank

Parametertyp
Konfigurierbar

Standardwert [Bereich]
No [Yes, No]

Wenn Sie diesen Parameter auf "Yes" setzen, müssen Sie zunächst ein vollständiges Backup der Datenbank durchführen, damit Sie über eine Ausgangsbasis für inkrementelle Backups verfügen. Wenn dieser Parameter aktiviert ist und ein Tabellenbereich erstellt wird, müssen Sie außerdem ein Backup erstellen, das den betreffenden Tabellenbereich umfasst. Dabei kann es sich entweder um ein Datenbank-Backup oder ein Tabellenbereichs-Backup handeln. Nach erfolgtem Backup dürfen inkrementelle Backups diesen Tabellenbereich umfassen.

tsm_mgmtclass - Tivoli Storage Manager-Verwaltungsklasse

Die Tivoli Storage Manager-Verwaltungsklasse (TSM) legt fest, wie der TSM-Server die Backup-Versionen der Objekte verwalten soll, die gesichert werden.

Konfigurationstyp
Datenbank

Parametertyp

- Online konfigurierbar
- Nach Member in einer DB2 pureScale-Umgebung konfigurierbar

Standardwert [Bereich]
NULL [beliebige Zeichenfolge]

Standardmäßig ist für DB2 keine Verwaltungsklasse angegeben.

Beim Ausführen eines TSM-Backups versucht TSM vor Verwendung der im Datenbankkonfigurationsparameter angegebenen Verwaltungsklasse zunächst, das Backup-Objekt an die Verwaltungsklasse zu binden, die in der Liste INCLUDE-EXCLUDE angegeben ist, die sich in der Datei mit den TSM-Clientoptionen befindet. Wird keine Übereinstimmung gefunden, wird die standardmäßige TSM-Verwaltungsklasse verwendet, die auf dem TSM-Server angegeben ist. Anschließend bindet TSM das Backup-Objekt erneut an die mit dem Datenbankkonfigurationsparameter angegebene Verwaltungsklasse.

Daher muss sowohl die Standardverwaltungsklasse als auch die mit dem Datenbankkonfigurationsparameter angegebene Verwaltungsklasse eine Backup-Kopiengruppe enthalten, da die Backup-Operation sonst fehlschlägt.

tsm_nodename - Tivoli Storage Manager-Knotenname ()

Mit diesem Parameter kann die Standardeinstellung für den Knotennamen, der dem Produkt Tivoli Storage Manager (TSM) zugeordnet ist, überschrieben werden.

Konfigurationstyp
Datenbank

Parametertyp

- Online konfigurierbar

- Nach Member in einer DB2 pureScale-Umgebung konfigurierbar

Weitergabeklasse
Anweisungsgrenzwert

Standardwert [Bereich]
NULL [beliebige Zeichenfolge]

Der Knotenname ist erforderlich, um eine Datenbank wiederherstellen zu können, die von einem anderen Knoten aus in TSM gesichert wurde.

Standardmäßig können Sie eine Datenbank in TSM nur auf dem Knoten wiederherstellen, von dem aus das Backup ausgeführt wurde. Es ist möglich, dass der Wert für den Parameter **tsm_nodename** bei einem Backup mit DB2 (z. B. mit dem Befehl **BACKUP DATABASE**) überschrieben wird.

tsm_owner - Tivoli Storage Manager-Eignername ()

Mit diesem Parameter kann die Standardeinstellung für den Eigner, der dem Produkt Tivoli Storage Manager (TSM) zugeordnet ist, überschrieben werden.

Konfigurationstyp
Datenbank

Parametertyp

- Online konfigurierbar
- Nach Member in einer DB2 pureScale-Umgebung konfigurierbar

Weitergabeklasse
Anweisungsgrenzwert

Standardwert [Bereich]
NULL [beliebige Zeichenfolge]

Der Eignername ist erforderlich, um eine Datenbank wiederherstellen zu können, die von einem anderen Knoten aus in TSM gesichert wurde. Es ist möglich, dass der Wert für den Parameter **tsm_owner** bei einem Backup mit DB2 (z. B. mit dem Befehl **BACKUP DATABASE**) überschrieben wird.

Anmerkung: Beim Eignernamen muss die Groß-/Kleinschreibung beachtet werden.

Standardmäßig können Sie eine Datenbank in TSM nur auf dem Knoten wiederherstellen, von dem aus das Backup ausgeführt wurde.

tsm_password - Tivoli Storage Manager-Kennwort ()

Mit diesem Parameter kann die Standardeinstellung für das Kennwort, das dem Produkt Tivoli Storage Manager (TSM) zugeordnet ist, überschrieben werden.

Konfigurationstyp
Datenbank

Parametertyp

- Online konfigurierbar
- Nach Member in einer DB2 pureScale-Umgebung konfigurierbar

Weitergabeklasse
Anweisungsgrenzwert

Standardwert [Bereich]
NULL [beliebige Zeichenfolge]

Dieses Kennwort ist erforderlich, um eine Datenbank wiederherstellen zu können, die in TSM von einem anderen Knoten aus gesichert wurde.

Anmerkung: Wenn der Parameter **tsm_nodename** bei einem Backup mit DB2 (z. B. mit dem Befehl **BACKUP DATABASE**) überschrieben wird, muss möglicherweise auch der Parameter **tsm_password** festgelegt werden.

Standardmäßig können Sie eine Datenbank in TSM nur auf dem Knoten wiederherstellen, von dem aus das Backup ausgeführt wurde. Es ist möglich, dass der Wert für den Parameter **tsm_nodename** bei einem Backup mit DB2 überschrieben wird.

user_exit_status - Statusanzeiger für Benutzerexit

Wenn der Parameter **user_exit_status** auf den Wert YES eingestellt wird, bedeutet dies, dass der Datenbankmanager für die aktualisierende Recovery aktiviert ist und dass die Datenbank Protokolldateien auf der Basis der Werte archiviert und abruft, die mit dem Parameter **logarchmeth1** oder **logarchmeth2** festgelegt wurden.

Konfigurationstyp
Datenbank

Parametertyp
Informativ

util_heap_sz - Zwischenspeichergröße für Dienstprogramme

Dieser Parameter gibt die maximale Speichergröße an, die gleichzeitig von den Dienstprogrammen BACKUP, RESTORE und LOAD (einschließlich LOAD RECOVERY) verwendet werden kann.

Konfigurationstyp
Datenbank

Parametertyp

- Online konfigurierbar
- Nach Member in einer DB2 pureScale-Umgebung konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
5000 [16 - 524 288]

Anmerkung: Der Standardwert unterliegt nach der ursprünglichen Datenbankerstellung Änderungen durch den DB2 Configuration Advisor.

Maßeinheit
Seiten (4 KB)

Zuordnung
Wenn für die Dienstprogramme des Datenbankmanagers erforderlich

Freigabe
Wenn der Speicher nicht mehr vom Dienstprogramm benötigt wird

Empfehlung: Verwenden Sie den Standardwert, bis Ihren Dienstprogrammen nicht mehr genügend Speicher zur Verfügung steht. Wenn der Speicher nicht ausreicht,

müssen Sie den Wert erhöhen. Wenn der auf Ihrem System verfügbare Speicher eingeschränkt ist, können Sie den Wert für diesen Parameter herabsetzen, um den von den Dienstprogrammen der Datenbank verwendeten Speicher zu begrenzen. Wenn ein zu niedriger Parameterwert angegeben wird, können Sie Dienstprogramme eventuell nicht gleichzeitig ausführen. Sie sollten diesen Parameter nach Bedarf dynamisch aktualisieren. Für eine kleine Anzahl von Dienstprogrammen setzen Sie diesen Parameter auf einen kleinen Wert. Für eine große Anzahl von Dienstprogrammen oder für speicherintensive Dienstprogramme sollten Sie diesen Parameter auf einen größeren Wert setzen.

varchar2_compat - VARCHAR2-Kompatibilität (Datenbankkonfigurationsparameter)

Dieser Parameter gibt an, ob die Kompatibilitätssemantik, die mit den Datentypen VARCHAR2 und NVARCHAR2 verknüpft ist, auf die verbundene Datenbank angewendet wird.

Konfigurationstyp
Datenbank

Parametertyp
Informativ

Der Wert wird bei der Erstellung der Datenbank festgelegt und basiert auf der Einstellung der Registrierdatenbankvariablen **DB2_COMPATIBILITY_VECTOR** für die Unterstützung des Datentyps VARCHAR2. Der Wert kann nicht geändert werden.

vendoropt - Lieferantenoptionen

Mit diesem Parameter werden zusätzliche Parameter angegeben, die DB2 möglicherweise zur Kommunikation mit Speichersystemen bei Backup-, Restore- oder Ladekopieoperationen benötigt.

Konfigurationstyp
Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
NULL [ ]

Einschränkungen
Mit dem Konfigurationsparameter **vendoropt** können keine herstellerspezifischen Optionen für Backup- oder Restoremomentaufnahmeoperationen angegeben werden. Zu diesem Zweck müssen Sie den Parameter **OPTIONS** der Backup- bzw. Restoredienstprogramme verwenden.

In Umgebungen mit Tivoli Storage Manager (TSM), die für die Unterstützung von Proxy-Knoten konfiguriert sind, sind die Optionen `"-fromnode=nodename"` und `"-fromowner=ownername"` nicht mit der Option `"-asnodename=nodename"` kompatibel und können nicht zusammen verwendet werden. Verwenden Sie die Option `-asnodename` für TSM-Konfigurationen, in denen Proxy-Knoten eingesetzt werden, und die beiden anderen

Optionen für andere Typen von TSM-Konfigurationen. Weitere Informationen finden Sie in „Konfigurieren eines Tivoli Storage Manager-Clients".

wlm_collect_int - Workload-Management-Erfassungsintervall (Konfigurationsparameter)

Dieser Parameter gibt ein Erfassungs- und Zurücksetzungsintervall (in Minuten) für die Workload-Management-Statistik an.

Alle *x* Minuten (wobei *x* der Wert des Parameters **wlm_collect_int** ist) werden alle Workload-Management-Statistikdaten gesammelt und an sämtliche aktiven Ereignismonitore für Statistiken gesendet. Anschließend wird die Statistik zurückgesetzt. Wenn ein aktiver Statistikereignismonitor vorhanden ist, wird die Statistik in Abhängigkeit davon, wie er erstellt wurde, in eine Datei, in eine Pipe oder in eine Tabelle geschrieben. Wenn kein aktiver Ereignismotor vorhanden ist, wird die Statistik nur zurückgesetzt, aber nicht erfasst.

Erfassungen erfolgen in den angegebenen Intervallzeiten relativ zum Zeitpunkt Sonntag 00:00:00 Uhr. Wenn das Katalogmember aktiv wird, erfolgt die nächste Erfassung beim Start des nächsten geplanten Intervalls relativ zu diesem festen Zeitpunkt. Das geplante Intervall ist nicht relativ zur Aktivierungszeit des Katalogmembers. Wenn ein Member zum Zeitpunkt der Erfassung nicht aktiv ist, werden keine Statistiken für dieses Member gesammelt. Wenn das Intervall zum Beispiel auf den Wert 60 gesetzt wurde und das Katalogmember am Sonntag um 09:24 Uhr aktiviert wurde, werden die Erfassungen zur Ausführung jede Stunde zur vollen Stunde geplant. Das heißt, die nächste Erfassung erfolgt um 10:00 Uhr. Wenn das Member um 10:00 Uhr nicht aktiv ist, werden für es keine Statistiken gesammelt.

Der Erfassungs- und Zurücksetzungsprozess wird vom Katalogmember eingeleitet. Der Parameter **wlm_collect_int** muss im Katalogmember angegeben werden. Er wird in keinem anderen Member verwendet.

Konfigurationstyp
Datenbank

Parametertyp

- Online konfigurierbar

Standardwert [Bereich]
`0` [`0` (keine Erfassung), `5 - 32 767`]

Die vom Ereignismonitor für Statistiken erfasste Workload-Management-Statistik kann zum Überwachen des Kurz- und Langzeitsystemverhaltens verwendet werden. Ein kleines Intervall kann zur Ermittlung des Kurz- und Langzeitsystemverhaltens verwendet werden, da die Ergebnisse zusammengefasst werden können, um ein Langzeitverhalten zu erhalten. Die erforderliche manuelle Zusammenfassung der Ergebnisse aus verschiedenen Intervallen verkompliziert allerdings die Analyse. Wenn es nicht erforderlich ist, führt ein kleines Intervall zu einer unnötigen Erhöhung der Verarbeitungszeit. Aus diesem Grund sollten Sie das Intervall verkleinern, um das Kurzzeitverhalten zu erfassen, und das Intervall vergrößern, um die Verarbeitungszeit zu reduzieren, wenn eine Analyse des Langzeitverhaltens bereits ausreicht.

Das Intervall muss jeweils für eine Datenbank, nicht für die einzelnen SQL-Anforderungen, Befehlsaufrufe oder Anwendungen, angepasst werden. Andere Konfigurationsparameter müssen nicht berücksichtigt werden.

Anmerkung: Alle WLM-Statistiktabellenfunktionen geben Statistikdaten zurück, die seit der letzten Zurücksetzung der Statistik gesammelt wurden. Die Statistik wird regelmäßig auf der Basis des durch diesen Konfigurationsparameter angegebenen Intervalls zurückgesetzt.

Konfigurationsparameter des DB2-Verwaltungsservers (DAS)

authentication - DAS-Authentifizierungstyp

Mit diesem Parameter wird festgelegt, wie und wo die Authentifizierung eines Benutzers stattfindet.

Wichtig: Der DB2-Verwaltungsserver (DAS) gilt in Version 9.7 als veraltet und wird möglicherweise in einem zukünftigen Release entfernt. Der DAS wird in DB2 pureScale-Umgebungen nicht unterstützt. Verwenden Sie Softwareprogramme, die das Secure Shell-Protokoll für die Fernverwaltung nutzen. Weitere Informationen hierzu finden Sie im Abschnitt „ DB2-Verwaltungsserver (DAS) gilt als veraltet" in http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0059276.html.

Konfigurationstyp
DB2-Verwaltungsserver

Gilt für
DB2-Verwaltungsserver

Parametertyp
Konfigurierbar

Standardwert [Bereich]
SERVER_ENCRYPT [SERVER_ENCRYPT; KERBEROS_ENCRYPT]

Hat der Parameter **authentication** den Wert SERVER_ENCRYPT, werden die Benutzer-ID und das Kennwort vom Client an den Server gesendet, damit die Authentifizierung auf dem Server ausgeführt werden kann. Über das Netz gesendete Benutzer-IDs und Kennwörter werden verschlüsselt.

Der Wert KERBEROS_ENCRYPT bedeutet, dass die Authentifizierung auf einem Kerberos-Server mithilfe des Kerberos-Sicherheitsprotokolls für Authentifizierung ausgeführt wird.

Anmerkung: Der Authentifizierungstyp KERBEROS_ENCRYPT wird nur auf Servern unterstützt, auf denen Windows ausgeführt wird.

Dieser Parameter kann nur über einen Befehlszeilenprozessor (CLP) der Version 9 aktualisiert werden.

contact_host - Speicherposition der Liste mit Ansprechpartnern

Dieser Parameter gibt die Speicherposition der Ansprechpartnerinformationen an, die der Scheduler und der Diagnosemonitor für Benachrichtigungen verwenden.

Konfigurationstyp
DB2-Verwaltungsserver

Gilt für
DB2-Verwaltungsserver

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
NULL [beliebiger gültiger TCP/IP-Hostname eines DB2-Verwaltungsservers]

Die Position muss der TCP/IP-Hostname eines DB2-Verwaltungsservers sein. Dadurch, dass sich der durch den Parameter **contact_host** angegebene Kontakthost auf einem fernen DAS befinden kann, wird eine gemeinsame Verwendung der Kontaktliste durch mehrere DB2-Verwaltungsserver unterstützt. Wenn **contact_host** nicht angegeben ist, nimmt der DAS an, dass die Kontaktinformationen lokal vorliegen.

Dieser Parameter kann nur über einen Befehlszeilenprozessor (CLP) der Version 8 aktualisiert werden.

das_codepage - DAS-Codepage

Dieser Parameter gibt die vom DB2-Verwaltungsserver verwendete Codepage an.

Wichtig: Der DB2-Verwaltungsserver (DAS) gilt in Version 9.7 als veraltet und wird möglicherweise in einem zukünftigen Release entfernt. Der DAS wird in DB2 pureScale-Umgebungen nicht unterstützt. Verwenden Sie Softwareprogramme, die das Secure Shell-Protokoll für die Fernverwaltung nutzen. Weitere Informationen hierzu finden Sie im Abschnitt „ DB2-Verwaltungsserver (DAS) gilt als veraltet” in http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0059276.html.

Konfigurationstyp
DB2-Verwaltungsserver

Gilt für
DB2-Verwaltungsserver

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
NULL [beliebige gültige DB2-Codepage]

Wenn der Parameter auf NULL gesetzt ist, wird die Standardcodepage des Systems verwendet. Dieser Parameter sollte mit der Ländereinstellung der lokalen DB2-Instanzen kompatibel sein. Andernfalls kann der DB2-Verwaltungsserver nicht mit den DB2-Instanzen kommunizieren.

Dieser Parameter kann nur über einen Befehlszeilenprozessor (CLP) der Version 8 aktualisiert werden.

das_territory - DAS-Gebiet

Dieser Parameter gibt das vom DB2-Verwaltungsserver verwendete Gebiet an.

Wichtig: Der DB2-Verwaltungsserver (DAS) gilt in Version 9.7 als veraltet und wird möglicherweise in einem zukünftigen Release entfernt. Der DAS wird in DB2 pureScale-Umgebungen nicht unterstützt. Verwenden Sie Softwareprogramme, die das Secure Shell-Protokoll für die Fernverwaltung nutzen. Weitere Informationen hierzu finden Sie im Abschnitt „ DB2-Verwaltungsserver (DAS) gilt als veraltet" in http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0059276.html.

Konfigurationstyp
DB2-Verwaltungsserver

Gilt für
DB2-Verwaltungsserver

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
NULL [beliebiges gültiges DB2-Gebiet]

Wenn der Parameter auf NULL gesetzt ist, wird das Standardgebiet des Systems verwendet.

Dieser Parameter kann nur über einen Befehlszeilenprozessor (CLP) der Version 8 aktualisiert werden.

dasadm_group - DASADM-Gruppenname

Dieser Parameter definiert den Gruppennamen mit DASADM-Berechtigung für den Datenbankverwaltungsserver.

Wichtig: Der DB2-Verwaltungsserver (DAS) gilt in Version 9.7 als veraltet und wird möglicherweise in einem zukünftigen Release entfernt. Der DAS wird in DB2 pureScale-Umgebungen nicht unterstützt. Verwenden Sie Softwareprogramme, die das Secure Shell-Protokoll für die Fernverwaltung nutzen. Weitere Informationen hierzu finden Sie im Abschnitt „ DB2-Verwaltungsserver (DAS) gilt als veraltet" in http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0059276.html.

Konfigurationstyp
DB2-Verwaltungsserver

Gilt für
DB2-Verwaltungsserver

Parametertyp
Konfigurierbar

Standardwert [Bereich]
Null [beliebiger gültiger Gruppenname]

Die Berechtigung DASADM ist die höchste Berechtigungsstufe innerhalb des Datenbankverwaltungsservers (DAS).

Die Berechtigung DASADM wird von den in einer bestimmten Betriebsumgebung verwendeten Sicherheitseinrichtungen ermittelt.

- Bei Windows-Betriebssystemen kann dieser Parameter auf eine beliebige lokale Gruppe gesetzt werden, die in der Sicherheitsdatenbank von Windows definiert

ist. Gruppennamen werden akzeptiert, solange sie höchstens eine Länge von 30 Byte haben. Wenn für diesen Parameter der Wert „NULL" definiert wird, haben alle Mitglieder der Administratorengruppe die Berechtigung DASADM.

- Wenn bei Linux- und UNIX-Systemen der Wert „NULL" für diesen Parameter angegeben wird, gilt die Primärgruppe des Instanzeigners standardmäßig als DASADM-Gruppe.

  Ist der Wert nicht „NULL", kann jeder gültige UNIX-Gruppenname als DASADM-Gruppe definiert werden.

db2system - Name des DB2-Serversystems

Dieser Parameter gibt den Namen an, der von Ihren Benutzern und Datenbankadministratoren verwendet wird, um das DB2-Serversystem anzugeben.

Konfigurationstyp
DB2-Verwaltungsserver

Gilt für
DB2-Verwaltungsserver

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
TCP/IP-Hostname [beliebiger gültiger Systemname]

Dieser Name sollte nach Möglichkeit innerhalb Ihres Netzes eindeutig sein.

Dieser Name unterstützt Benutzer bei der Identifizierung des Systems, das die Datenbank enthält, auf die sie zugreifen möchten. Bei der Installation wird für **db2system** wie folgt ein Wert festgelegt:

- Unter Windows setzt das Installationsprogramm (**setup**) diesen Parameter auf den Wert des Computernamens, der für das Windows-System angegeben ist.
- Auf UNIX-Systemen wird für diesen Parameter der TCP/IP-Hostname des UNIX-Systems definiert.

diaglevel - Aufzeichnungsebene bei Fehlerdiagnose (Konfigurationsparameter)

Durch diesen Parameter wird der Typ der Diagnosefehler festgelegt, die in der Datei `db2dasdiag.log` aufgezeichnet werden.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Client
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
`3 [0 - 4]`

Gültige Werte für diesen Parameter sind:

- `0` – Keine Aufzeichnung von Diagnosedaten
- `1` – Nur schwerwiegende Fehler
- `2` – Alle Fehler
- `3` – Alle Fehler und Warnungen
- `4` – Alle Fehler, Warnungen und Informationsnachrichten

Hinweise

- Das dynamische Verhalten für **diaglevel** erstreckt sich nicht auf alle Prozesse.
- Der DB2-Serverprozess **db2sysc** kann dynamische Änderungen erkennen, z. B. wenn der Befehl **UPDATE DATABASE MANAGER CONFIGURATION** für eine Instanzzuordnung ausgegeben wird.
- Wenn DB2-Client- und -Anwendungsprozesse gestartet werden, wird dafür die Einstellung des Konfigurationsparameters **diaglevel** verwendet und es werden keine dynamischen Änderungen erkannt.
- Zum Lösen eines Problems können Sie den Wert dieses Parameters erhöhen, um zusätzliche Fehlerbestimmungsdaten zu sammeln.
- In bestimmten Situationen überschreibt DB2 die Einstellung des Konfigurationsparameters **diaglevel**, um Nachrichten mit hohem Stellenwert anzuzeigen.

discover - DAS-Discovery-Modus

Dieser Parameter bestimmt den Typ des Discovery-Modus, der beim Start des DB2-Verwaltungsservers gestartet wird.

Wichtig: Der DB2-Verwaltungsserver (DAS) gilt in Version 9.7 als veraltet und wird möglicherweise in einem zukünftigen Release entfernt. Der DAS wird in DB2 pureScale-Umgebungen nicht unterstützt. Verwenden Sie Softwareprogramme, die das Secure Shell-Protokoll für die Fernverwaltung nutzen. Weitere Informationen hierzu finden Sie im Abschnitt „ DB2-Verwaltungsserver (DAS) gilt als veraltet” in http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0059276.html.

Konfigurationstyp
DB2-Verwaltungsserver

Gilt für
DB2-Verwaltungsserver

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
`SEARCH [DISABLE; KNOWN; SEARCH ]`

- Wenn **discover** = `SEARCH` angegeben wird, bearbeitet der Verwaltungsserver Discovery-Anforderungen von Clients mit dem Parameterwert SEARCH. SEARCH stellt eine Obermenge der durch Discovery mit dem Parameter KNOWN bereitgestellten Funktionalität zur Verfügung. Wenn **discover** = `SEARCH` angegeben wird, bearbeitet der Verwaltungsserver Discovery-Anforderungen von Clients sowohl mit dem Parameterwert SEARCH als auch mit dem Parameterwert KNOWN.
- Wenn **discover** = `KNOWN` angegeben wird, bearbeitet der Verwaltungsserver nur Discovery-Anforderungen von Clients mit dem Parameterwert KNOWN.

- Wenn **discover** = DISABLE angegeben wird, bearbeitet der Verwaltungsserver keine Discovery-Anforderungen. Die Informationen für dieses Serversystem sind im Wesentlichen vor Clients verdeckt.

Der Discovery-Modus ist standardmäßig aktiviert.

exec_exp_task - Verfallene Tasks ausführen

Dieser Parameter gibt an, ob der Scheduler Tasks ausführt, die in der Vergangenheit terminiert, bislang jedoch noch nicht ausgeführt wurden.

Konfigurationstyp
DB2-Verwaltungsserver

Gilt für
DB2-Verwaltungsserver

Parametertyp
Konfigurierbar

Standardwert [Bereich]
No [Yes; No]

Der Scheduler erkennt verfallene Tasks nur dann, wenn er gestartet wird. Wenn Sie beispielsweise die Ausführung eines Jobs für jeden Samstag terminiert haben und der Scheduler am Freitag ausgeschaltet und am Montag erneut gestartet wird, ist der für Samstag terminierte Job nun ein in der Vergangenheit terminierter Job. Wenn der Parameter **exec_exp_task** auf Yes gesetzt ist, wird der für Samstag terminierte Job beim Start des Schedulers ausgeführt.

jdk_path - Installationspfad für Software Developer's Kit für Java auf DAS

Dieser Parameter gibt das Verzeichnis an, in dem das Software Developer's Kit (SDK) für Java installiert ist, das zu verwenden ist, um DB2-Verwaltungsserverfunktionen auszuführen.

Konfigurationstyp
DB2-Verwaltungsserver

Gilt für
DB2-Verwaltungsserver

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
Java-Standardinstallationspfad [beliebiger gültiger Pfad]

Die vom Java-Interpreter verwendeten Umgebungsvariablen werden aus dem Wert dieses Parameters errechnet.

Unter Windows-Betriebssystemen werden Java-Dateien (falls erforderlich) während der DB2-Installation im Verzeichnis sqllib (in java\jdk) gespeichert. Der Konfigurationsparameter **jdk_path** wird anschließend auf sqllib\java\jdk festgelegt. Java wird tatsächlich nicht durch DB2 auf Windows-Plattformen installiert. Die Dateien werden lediglich im Verzeichnis sqllib abgelegt, und zwar unabhängig davon, ob Java bereits installiert ist.

Dieser Parameter kann nur über einen Befehlszeilenprozessor (CLP) der Version 8 aktualisiert werden.

sched_enable - Schedulermodus

Mit diesem Parameter wird angegeben, ob der Scheduler vom Verwaltungsserver gestartet wird oder nicht.

Konfigurationstyp
DB2-Verwaltungsserver

Gilt für
DB2-Verwaltungsserver

Parametertyp
Konfigurierbar

Standardwert [Bereich]
Off [On; Off]

Der Scheduler ermöglicht Tools wie Data Studio das Terminieren und Ausführen von Tasks auf dem Verwaltungsserver.

Dieser Parameter kann nur über einen Befehlszeilenprozessor (CLP) der Version 8 aktualisiert werden.

sched_userid - Benutzer-ID für Scheduler

Dieser Parameter gibt die Benutzer-ID an, die der Scheduler verwendet, um eine Verbindung zur Toolskatalogdatenbank herzustellen. Dieser Parameter ist nur dann relevant, wenn die Toolskatalogdatenbank eine ferne Datenbank für den DB2-Verwaltungsserver ist.

Konfigurationstyp
DB2-Verwaltungsserver

Gilt für
DB2-Verwaltungsserver

Parametertyp
Informativ

Standardwert [Bereich]
Null [beliebige gültige Benutzer-ID]

Die vom Scheduler für eine Verbindung zur fernen Toolskatalogdatenbank verwendete Benutzer-ID und das Kennwort werden mit dem Befehl **db2admin** angegeben.

smtp_server - SMTP-Server

Wenn der Scheduler aktiviert ist, gibt dieser Parameter den SMTP-Server an, den der Scheduler zum Versenden von E-Mail- und Pagerbenachrichtigungen verwendet.

Konfigurationstyp
DB2-Verwaltungsserver

Gilt für
DB2-Verwaltungsserver

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
NULL [beliebiger gültiger TCP/IP-Hostname für SMTP-Server]

Dieser Parameter wird vom Scheduler und dem Diagnosemonitor verwendet.

Dieser Parameter kann nur über einen Befehlszeilenprozessor (CLP) der Version 8 aktualisiert werden.

toolscat_db - Toolskatalogdatenbank

Dieser Parameter gibt die vom Scheduler verwendete Toolskatalogdatenbank an.

Konfigurationstyp
DB2-Verwaltungsserver

Gilt für
DB2-Verwaltungsserver

Parametertyp
Konfigurierbar

Standardwert [Bereich]
Null [beliebiger gültiger Aliasname der Datenbank]

Diese Datenbank muss sich im Datenbankverzeichnis der Instanz befinden, das vom Parameter **toolscat_inst** angegeben ist.

toolscat_inst - Instanz der Toolskatalogdatenbank

Dieser Parameter gibt den Instanznamen an, der vom Scheduler zusammen mit 'toolscat_db' und 'toolscat_schema' zur Identifikation der Toolskatalogdatenbank verwendet wird.

Konfigurationstyp
DB2-Verwaltungsserver

Gilt für
DB2-Verwaltungsserver

Parametertyp
Konfigurierbar

Standardwert [Bereich]
NULL [beliebige gültige Instanz]

Die Toolskatalogdatenbank enthält Taskinformationen. Diese Toolskatalogdatenbank muss sich im Datenbankverzeichnis der Instanz befinden, das von diesem Konfigurationsparameter angegeben wird. Bei der Datenbank kann es sich um eine lokale oder ferne Datenbank handeln. Bei Toolskatalogdatenbank muss die Instanz für TCP/IP konfiguriert sein. Bei einer fernen Datenbank muss die Datenbankpartition, die im Datenbankverzeichnis katalogisiert wird, ein TCP/IP-Knoten sein.

toolscat_schema - Schema der Toolskatalogdatenbank

Dieser Parameter gibt das Schema der vom Scheduler verwendeten Toolskatalogdatenbank an.

Konfigurationstyp
DB2-Verwaltungsserver

Gilt für
DB2-Verwaltungsserver

Parametertyp
Konfigurierbar

Standardwert [Bereich]
Null [beliebiges gültiges Schema]

Das Schema wird verwendet, um eine Reihe von Toolskatalogtabellen und -sichten innerhalb der Datenbank eindeutig anzugeben.

Dieser Parameter kann nur über einen Befehlszeilenprozessor (CLP) der Version 8 aktualisiert werden.

CF_sca_sz - Gemeinsamer Kommunikationsbereich (Konfigurationsparameter)

Dieser Konfigurationsparameter für den gemeinsamen Kommunikationsbereich (Shared Communication Area, SCA) legt fest, welche Speicherkapazität von SCA in der Cluster-Caching-Funktion (CF) verwendet wird. Der gemeinsame Kommunikationsbereich (SCA) ist eine Einheit auf Datenbankebene, die datenbankweite Steuerblockdaten für Tabellen, Indizes, Tabellenbereiche und Kataloge enthält.

Konfigurationstyp
Datenbank

Gilt für
DB2 pureScale-Umgebung

Parametertyp
Online konfigurierbar

Standardwert [Bereich]
AUTOMATIC [AUTOMATIC, 2048 – 1073741824]

Maßeinheit
Seiten (4 KB)

Zuordnung
Bei der ersten Aktivierung der Datenbank auf einem DB2-Member

Freigabe
Wenn die Datenbank gelöscht oder der CF-Server gestoppt wird

Wenn Sie den Parameter **cf_sca_sz** auf den Standardwert AUTOMATIC setzen, wird beim ersten Aktivieren der Datenbank auf einem beliebigen Member vom DB2-Datenbankmanager ein Speicherwert berechnet, der für grundlegende Datenbankoperationen ausreicht. Wenn Sie die genaue Speichergröße für den SCA konfigurieren möchten, können Sie den Parameter **cf_sca_sz** mit einem festen numerischen Wert definieren.

Den aktuellen Wert für die SCA-Größe können Sie durch Ausführen des Befehls **GET DB CFG SHOW DETAIL** abrufen. Wenn der Parameter **cf_sca_sz** auf AUTOMATIC gesetzt ist, zeigt die Klausel SHOW DETAIL den berechneten Wert sowie den zugeordneten Wert für den SCA an.

Wenn der SCA-Speicher durch Datenbankoperationen vollständig ausgelastet ist, wird eine Fehlernachricht für Speicherengpass zurückgegeben. In diesem Fall können Sie den SCA-Speicher vergrößern, indem Sie einen höheren Wert für den Parameter **cf_sca_sz** angeben.

Weitere Details finden Sie in „Speicher der Cluster-Caching-Funktion für eine Datenbank konfigurieren".

DB2 pureScale Feature - Konfiguration der Cluster-Caching-Funktion

In einer DB2 pureScale-Umgebung wird die Cluster-Caching-Funktion (auch als CF bezeichnet) verwendet, um die globale Sperren- und Pufferpoolverwaltung zu vereinfachen. Die CF-Konfiguration ist in die DB2-Infrastruktur integriert und wird über DB2-Konfigurationsschnittstellen verwaltet.

Die CF-Konfiguration verfügt über spezifische Konfigurationsparameter zur Optimierung der DB2 pureScale-Umgebung.

Konfigurationsparameter für die CF werden entweder als CF-Serverkonfigurationsparameter oder als CF-Strukturkonfigurationsparameter kategorisiert. CF-Serverparameter werden als Konfigurationsparameter des DB2-Datenbankmanagers konfiguriert. CF-Strukturparameter werden als DB2-Datenbankkonfigurationsparameter konfiguriert. Auf die Konfigurationsdatei werden Standardeinstellungen angewendet, wenn eine DB2 pureScale Feature-Instanz oder eine Datenbank erstellt wird.

Konfiguration der Cluster-Caching-Funktion

Die Informationen der CF-Serverkonfiguration werden als Teil der Konfigurationsinformationen für den DB2-Datenbankmanager verwaltet. An sich wird der CF-Server ebenso über DB2-Konfigurationsschnittstellen konfiguriert wie der Datenbankmanager. Zum Anzeigen und Aktualisieren von CF-Serverinformationen können der DB2-Befehlszeilenprozessor (CLP) oder die Konfigurations-API verwendet werden.

Die folgenden Konfigurationsparameter unterstützen den CF-Server:

- **cf_diaglevel**
- **cf_diagpath**
- **cf_mem_sz**
- **cf_num_conns**
- **cf_num_workers**

Strukturkonfiguration der Cluster-Caching-Funktion

Der CF-Strukturspeicher wird als Teil der Informationen zur DB2-Datenbankkonfiguration verwaltet. CF-Strukturparameter werden ebenso wie DB2-Datenbankparameter über den DB2-Befehlszeilenprozessor (CLP) oder über Konfigurations-APIs zum Anzeigen und Aktualisieren der konfigurierbaren CF-Struktur konfiguriert.

Zur Unterstützung der CF-Struktur wurden die folgenden Konfigurationsparameter hinzugefügt:

- **cf_db_mem_sz**
- **cf_gbp_sz**
- **cf_lock_sz**
- **cf_sca_sz**

CF- Strukturen werden im CF-Server für jede Datenbank erstellt. Während der Befehl **CREATE DATABASE** aufgerufen wird, setzt der DB2-Konfigurationsadvisor auch die CF-Strukturkonfigurationsparameter auf ihren Standardwert AUTOMATIC und berechnet die passenden Startwerte für Ihre DB2 pureScale-Instanz.

Die richtige CF-Strukturkonfiguration ist wichtig, um optimale Leistung in einer DB2 pureScale-Umgebung zu erzielen. Mit den berechneten Standardwerten für CF-Strukturen lässt sich unter Umständen nicht die optimale Leistung für Ihre DB2 pureScale-Umgebung erzielen. Führen Sie in einer Leistungstestumgebung eine manuelle Optimierung der Größen für die CF-Struktur durch, um das erforderliche Leistungsniveau zu realisieren. Sie müssen auch daran denken, dass die Verwendung der Tabellenpartitionierung unter DB2 pureScale zu erhöhter Speicherbelegung führt.

Konfigurationsadvisor

Der DB2-Konfigurationsadvisor wird bei der Erstellung einer Datenbank aufgerufen, um einen angemessenen Satz von Standardwerten für die Datenbankkonfiguration in der Hostumgebung festzulegen. Dazu gehören zum Beispiel die Speicherkapazität und die Anzahl der Prozessoren.

Der DB2-Konfigurationsadvisor wird zur Berechnung der Startwerte für die CF-Strukturkonfigurationsparameter durch das DB2 pureScale Feature erweitert. Die Parameter **cf_db_mem_sz**, **cf_gbp_sz**, **cf_lock_sz** und **cf_sca_sz** werden auf der Basis der Größe des CF-Gesamtspeichers (definiert durch den Parameter **cf_mem_sz**) berechnet, um die beste Verteilung des Speichers unter diesen Strukturen sicherzustellen, damit ein angemessenes Leistungsniveau für eine DB2 pureScale-Instanz erzielt werden kann.

Hohe Verfügbarkeit mit zwei Cluster-Caching-Funktionen

Sie müssen zwei CF-Server einrichten, wenn mit einer hoch verfügbaren CF-Umgebung gearbeitet wird. Die eine CF fungiert als primärer Server und die andere als Backup- oder sekundärer CF-Server. In einer solchen Umgebung haben der primäre CF-Server und der sekundäre CF-Server dieselbe Hostkonfiguration.

Weitere Informationen zur Duplexunterstützung der Struktur in Hochverfügbarkeitsumgebungen finden Sie im Abschnitt zum Verhalten der Duplexunterstützung der Struktur mit einer sekundären Cluster-Caching-Funktion.

In einer DB2 pureScale-Umgebung mit zwei CF-Servern können Sie beide Cluster-Caching-Funktionen ersetzen, indem Sie ein schrittweises Upgrade ausführen, um eine hoch verfügbare Umgebung beizubehalten.

Weitere Informationen zur Beibehaltung einer Hochverfügbarkeitsumgebung beim Ersetzen von CFen finden Sie im Abschnitt zum Ersetzen beider Cluster-Caching-Funktionen.

Konfigurieren der Cluster-Caching-Funktion

Verwenden Sie den DB2-Befehlszeilenprozessor (CLP) oder die DB2-Konfigurations-APIs, um die Parameter der Cluster-Caching-Funktion (CF) für IBM DB2 pureScale Feature zu konfigurieren.

Informationen zu diesem Vorgang

Die Konfigurationsinformationen für die Cluster-Caching-Funktion werden als Teil der Konfigurationsinformationen für den DB2-Datenbankmanager verwaltet.

Wenn die Cluster-Caching-Funktion gestartet wird, werden die Konfigurationsparameter für den CF-Server aus der Konfigurationsdatei des Datenbankmanagers gele-

sen, um den CF-Server zu initialisieren. Der CF-Server erstellt einen Management-Port, über den er für CF-Serverkonfigurationsdaten empfangsbereit ist. Sobald ein Befehl **db2start** oder eine Onlineaktion durch einen CF-Server oder eine CF-Struktur abgeschlossen ist, empfängt der CF-Server die Serverkonfigurationsparameter und wird vollständig initialisiert.

Wenn die Konfigurationsparameter des CF-Servers auf AUTOMATIC gesetzt sind, werden die Werte automatisch während des Starts des CF-Servers bestimmt. Während des Starts wird der CF-Server-Host nach dem verfügbaren physischen Arbeitsspeicher und der Anzahl der CPUs abgefragt. Dieser Vorgang hilft dabei, die richtigen Werte für die Konfigurationsparameter **cf_mem_sz** und **cf_num_workers** festzulegen.

Es gibt zwei Möglichkeiten, den CF-Server zu konfigurieren:

- Aktualisieren von Konfigurationsparametern über den Befehlszeilenprozessor (CLP)
- Aktualisieren von Konfigurationsparametern über APIs

Vorgehensweise

- Zum Anzeigen der Werte der Konfigurationsparameter für den CF-Server über den Befehlszeilenprozessor geben Sie den Befehl **GET DATABASE MANAGER CONFIGURATION** ein.

  Beispielsweise werden die nachfolgenden Informationen für die CF-Serverkonfiguration angezeigt, wenn der folgende Befehl in einer DB2 pureScale-Instanz abgesetzt wird:

  ```
  CF-Serverkonfiguration:
     Speichergröße (4 KB)                      (CF_MEM_SZ) = AUTOMATIC
     Anzahl Worker-Threads                (CF_NUM_WORKERS) = AUTOMATIC
     Anzahl Verbindungen                    (CF_NUM_CONNS) = AUTOMATIC
     Aufzeichnungsebene bei Fehlerdiagnose   (CF_DIAGLEVEL) = 2
     Verzeichnispfad für Diagnosedaten        (CF_DIAGPATH)
  ```

  Weitere Informationen mit detaillierteren Ausgaben finden Sie im Abschnitt zum Befehl **GET DATABASE MANAGER CONFIGURATION**.

- Zum Aktualisieren der Werte der Konfigurationsparameter für den CF-Server über den Befehlszeilenprozessor (CLP) geben Sie den Befehl **UPDATE DATABASE MANAGER CONFIGURATION** ein.

  Zum Beispiel werden solche Informationen wie die folgenden angezeigt, wenn dieser Befehl zum Aktualisieren der CF-Speichergröße (**cf_mem_sz**) eingegeben wird:

  ```
  UPDATE DBM CFG USING CF_MEM_SZ 27152

  DB20000I  Der Befehl UPDATE DATABASE MANAGER CONFIGURATION wurde erfolgreich ausgeführt.

  SQL1362W  Mindestens einer der zur sofortigen Änderung übergebenen Parameter wurde nicht
  dynamisch geändert. Clientänderungen werden erst beim nächsten Starten der Anwendung oder
  nach Absetzen des Befehls TERMINATE wirksam. Serveränderungen werden erst beim nächsten
  Absetzen des Befehls DB2START wirksam.
  ```

  Die Änderungen an der CF-Serverkonfiguration werden nicht sofort wirksam. Alle aktualisierten Werte werden gespeichert und beim nächsten Start des CF-Server angewendet.

- Zum Zurücksetzen der Werte der Konfigurationsparameter für den CF-Server über den Befehlszeilenprozessor (CLP) geben Sie den Befehl **RESET DATABASE MANAGER CONFIGURATION** (oder **RESET DBM CFG**) ein. Die Konfigurationsparameter für den DB2-Datenbankmanager werden auf ihren Standardwert zurückgesetzt.

- Wenn Sie die Werte der Konfigurationsparameter für den CF-Server mithilfe von APIs (Anwendungsprogrammierschnittstellen) aktualisieren wollen, rufen Sie die API db2CfgSet auf. Zum Anzeigen der Werte rufen Sie die API db2CfgGet auf.

  Das Abrufen und Festlegen von Konfigurationsparametern für den CF-Server wird von Einträgen für das Tokenelement in den **db2Cfg**-Datenstrukturen unterstützt, die beim Aufrufen der API db2CfgGet oder db2CfgSet verwendet werden.

*Tabelle 139. Unterstützte **db2Cfg**-Token für das Abrufen und Festlegen von Konfigurationsparametern für den CF-Server*

Token	Datentyp
SQLF_KTN_CF_MEM_SZ	Uint32
SQLF_KTN_CF_NUM_WORKERS	Uint32
SQLF_KTN_CF_NUM_CONNS	Uint32
SQLF_KTN_CF_DIAGLEVEL	Uint16
SQLF_KTN_CF_DIAGPATH SQLF_KTN_CF_DIAGPATH_FULL	char(215)

Konfigurieren des Speichers der Cluster-Caching-Funktion für eine Datenbank

Verwenden Sie den DB2-Befehlszeilenprozessor (CLP) oder die DB2-Konfigurations-APIs, um die Konfigurationsparameter für den Speicher der Cluster-Caching-Funktion für die Datenbank anzuzeigen und zu aktualisieren.

Informationen zu diesem Vorgang

Die Konfigurationsinformationen für den Strukturspeicher der Cluster-Caching-Funktion werden als Teil der Konfigurationsinformationen für die DB2-Datenbank verwaltet.

Der Strukturspeicher der Cluster-Caching-Funktion, der für den Gruppenpufferpool (GBP), die Sperrennutzung und den gemeinsamen Kommunikationsbereich (SCA) verwendet wird, wird für die Cluster-Caching-Funktion bei der ersten Datenbankaktivierung auf einem der Member zugeordnet und bleibt bis zur Inaktivierung der Datenbank auf dem letzten Member zugeordnet. Diese Parameter werden standardmäßig auf den Wert `AUTOMATIC` gesetzt. Für Parameter, die auf den Wert `AUTOMATIC` gesetzt sind, berechnet DB2 geeignete Werte während der Datenbankaktivierung. Da diese Werte eng zusammengehören und voneinander abhängig sind, bewirkt eine manuelle Einstellung mindestens eines der Parameter, dass keiner der Parameter bei der Datenbankaktivierung berechnet wird, selbst wenn einige Parameter den Wert `AUTOMATIC` beibehalten. Die Parameter behalten ihre zuletzt automatisch berechneten Werte, die mithilfe des Befehls **`GET DB CFG SHOW DETAIL`** angezeigt werden können.

Für Strukturparameter wird auch die Option `ONLINE` unterstützt. Alle Aktualisierungen an CF-Speicherparametern werden unverzüglich angewendet. Aktualisierungsanforderungen erfolgen synchron und erst zurückgegeben, wenn der neue Wert vom CF-Server eingestellt wurde.

Es gibt zwei Möglichkeiten, den Speicher der Cluster-Caching-Funktion für eine Datenbank zu konfigurieren:

- Aktualisieren von Konfigurationsparametern über den Befehlszeilenprozessor (CLP)

- Aktualisieren von Konfigurationsparametern über APIs

Vorgehensweise

- Zum Anzeigen der Werte der Konfigurationsparameter für die CF-Struktur über den Befehlszeilenprozessor geben Sie den Befehl **GET DATABASE CONFIGURATION** ein.

  Beispielsweise werden die nachfolgenden Informationen für die CF-Strukturkonfiguration angezeigt, wenn der folgende Befehl in einer DB2 pureScale-Instanz abgesetzt wird:

  ```
  Konfiguration der CF-Ressourcen:
       Größe des CF-Datenbankspeichers (4 KB)           (CF_DB_MEM_SZ) = AUTOMATIC
       Größe des Gruppenpufferpools (4 KB)                 (CF_GBP_SZ) = AUTOMATIC
       Speichergröße für globale Sperren (4 KB)          (CF_LOCK_SZ) = AUTOMATIC
       Größe des gemeinsamen Kommunikationsbereichs (4 KB) (CF_SCA_SZ) = AUTOMATIC
  ```

  Weitere Informationen mit detaillierteren Ausgaben finden Sie im Abschnitt zum Befehl **GET DATABASE CONFIGURATION**.

  Anmerkung: Der Parameter für den Gruppenpufferpool (GBP) erfordert eine ausreichende Anzahl von Verzeichniseinträgen in der CACHE-Struktur, um Lese- und Registrierungsanforderungen (RAR-Anforderungen, Read-and-Register) verarbeiten zu können. DB2 überwacht für jede RAR-Anforderung die freien Verzeichniseinträge und die Anzahl der Datenelemente und passt diese Werte entsprechend an. Diese Anpassung erfolgt dynamisch oder durch Onlineaktualisierungen, unabhängig davon, ob der Parameter **cf_gbp_sz** mit dem Wert AUTOMATIC konfiguriert ist oder nicht. Weitere Informationen zu neuen CF-Strukturkonfigurationsparameter finden Sie im Abschnitt "Konfigurationsparameter - Zusammenfassung für DB2 pureScale Feature".

  Wenn Sie zum Anzeigen der aktuellen Strukturkonfigurationsparameter die Option **SHOW DETAIL** verwenden, werden die folgenden Informationen angezeigt:

  ```
  Konfiguration der CF-Ressourcen:
    Größe des Gruppenpufferpools (4 KB)                 (CF_GBP_SZ) = AUTOMATIC(32768)
           AUTOMATIC(32768)
    Speichergröße für globale Sperren (4 KB)          (CF_LOCK_SZ) = AUTOMATIC(17000)
           AUTOMATIC(17000)
    Größe des gemeinsamen Kommunikationsbereichs (4 KB) (CF_SCA_SZ) = AUTOMATIC(1500)
           AUTOMATIC(1500)
  ```

  Weitere Informationen mit detaillierteren Ausgaben finden Sie im Abschnitt zum Befehl **GET DATABASE CONFIGURATION**.

  Das für die einzelnen Strukturparameter angezeigte Ergebnis hat zwei entsprechende Werte. Wenn sich die Werte in Klammern unterscheiden, steht eine Aktualisierung am Wert des Strukturparameters an. Der Wert in der ersten Klammer ist der aktuelle Wert für den Strukturparameter. Der Wert in der zweiten Klammer ist der zukünftige Wert für den Strukturparameter.

  Wenn sowohl ein primärer CF-Server als auch ein sekundärer CF-Server vorhanden ist, werden die Konfigurationsinformationen nur für den primären CF-Server angezeigt. Für beide CF-Server gilt derselbe Wert.

- Zum Aktualisieren der Werte der Konfigurationsparameter für die CF-Struktur über den Befehlszeilenprozessor (CLP) geben Sie den Befehl **UPDATE DATABASE CONFIGURATION** (oder **UPDATE DB CFG**) ein. Eine Aktualisierung von Strukturparametern auf einen anderen Wert als die Standardeinstellung (AUTOMATIC) bewirkt, dass die Speichergröße für die CF-Struktur angepasst wird.

  Dieser Befehl wird beispielsweise abgesetzt, um die Speichergröße des globalen Sperrenmanagers auf 20.000 Seiten zu setzen:

  ```
  UPDATE DATABASE CONFIGURATION FOR WSDB USING CF_LOCK_SZ 20000
  ```

Anmerkung: Eine Aktualisierung zum Verkleinern des Speichers für Strukturparameter wird nicht sofort umgesetzt, wenn nicht genügend ungenutzter Speicher zur Ausführung der Anforderung vorhanden ist. Führen Sie den Befehl **GET DB CFG** über den Befehlszeilenprozessor aus, um festzustellen, ob für Strukturparameter eine Aktualisierung ansteht.

- Zum Zurücksetzen der Werte der Konfigurationsparameter für die CF-Struktur über den Befehlszeilenprozessor (CLP) geben Sie den Befehl **RESET DATABASE CONFIGURATION** (oder **RESET DB CFG**) ein. Die Konfigurationsparameter für die CF-Struktur werden auf ihren Standardwert AUTOMATIC zurückgesetzt.
- Wenn Sie die Werte der Konfigurationsparameter für die CF-Struktur mithilfe von APIs (Anwendungsprogrammierschnittstellen) aktualisieren wollen, rufen Sie die API db2CfgSet auf. Zum Anzeigen der Werte rufen Sie die API db2CfgGet auf.

  Das Abrufen und Festlegen von Konfigurationsparametern für die CF-Struktur wird von Einträgen für das Tokenelement in den **db2Cfg**-Datenstrukturen unterstützt, die beim Aufrufen der API db2CfgGet oder db2CfgSet verwendet werden.

*Tabelle 140. Unterstützte **db2Cfg**-Token für das Abrufen und Festlegen von Konfigurationsparametern für die CF-Struktur*

Token	Datentyp
SQLF_DBTN_CF_GBP_SZ	Uint32
SQLF_DBTN_CF_SCA_SZ	Uint32
SQLF_DBTN_CF_LOCK_SZ	Uint32
SQLF_DBTN_CF_DB_MEM_SZ	Uint32

Konfiguration der CF-Speicherparameter für DB2 pureScale

In einer DB2 pureScale-Instanz steuern verschiedene Konfigurationsparameter und Registrierungsdatenbankvariablen gemeinsam die Speicherzuordnung für die Cluster-Caching-Funktion (auch CF genannt).

Parameter

Die folgenden Konfigurationsparameter und Registrierungsdatenbankvariablen beeinflussen die CF-Speicherzuordnung und -Verwaltung. Diese Parameter und Variablen werden erst nach dem Start oder Neustart eines Members wirksam (einige Ausnahmen finden Sie in den entsprechenden Beschreibungen weiter unten).

- **cf_mem_sz**: Steuert die Gesamtspeichergröße, die von der CF verwendet wird. Die Standardeinstellung ist AUTOMATIC.
- **cf_db_mem_sz**: Steuert das Gesamtspeicherlimit, das von der CF für diese bestimmte Datenbank verwendet wird. Die Standardeinstellung ist AUTOMATIC.
- **cf_gbp_sz**: Legt den Speicher fest, der von der CF für die Gruppenpufferpoolnutzung für diese bestimmte Datenbank verwendet wird. Die Standardeinstellung ist AUTOMATIC.
- **cf_lock_sz**: Legt den Speicher fest, der von der CF für das Sperren für diese bestimmte Datenbank verwendet wird. Die Standardeinstellung ist AUTOMATIC.
- **cf_sca_sz**: Legt den Speicher fest, der von dieser CF für den gemeinsamen Kommunikationsbereich (Shared Communication Area, SCA) für diese bestimmte Datenbank verwendet wird. Die Standardeinstellung ist AUTOMATIC.
- **numdb**: Gibt die Anzahl der lokalen Datenbank an, die gleichzeitig aktiv sein können. Die Standardanzahl der aktiven Datenbanken ist 32. Dieser Parameter wird erst nach einen Gruppenneustart wirksam.

- **DB2_DATABASE_CF_MEMORY**: Diese Registrierdatenbankvariable legt den Prozentsatz des CF-Gesamtspeichers fest, der allen Datenbanken zugeordnet ist, für die der Wert **cf_db_mem_sz** auf AUTOMATIC gesetzt ist. Der Standardprozentsatz ist 100. Diese Variable wird erst nach einen Gruppenneustart wirksam.

Wenn sie auf AUTOMATIC gesetzt sind, werden die Werte der CF-Speicherkonfigurationsparameter basierend auf den Eigenschaften und Attributen der DB2 pureScale-Umgebung dynamisch erstellt.

Parameterbeziehungen

Anmerkung: Geben Sie für den Parameter **numdb** einen Wert an, der größer oder gleich der Gesamtzahl der Datenbanken in der Instanz ist. Dieser Wert umfasst aktive und inaktive Datenbanken. Außerdem können Sie für **DB2_DATABASE_CF_MEMORY** einen Wert angeben, der zulässt, dass alle Datenbanken in dieser Instanz gleichzeitig aktiv sein können (unabhängig davon, ob sie normalerweise inaktiv oder aktiv sind).

Die Speicherparameter für die Cluster-Caching-Funktion werden sowohl auf Instanz- als auch auf Datenbankebene definiert:

- Auf Instanzebene wird die für die CF reservierte Gesamtspeichermenge durch den Konfigurationsparameter **cf_mem_sz** festgelegt.
- Auf Datenbankebene wird die von allen CF-Strukturen für eine Datenbank verwendete Gesamtspeichermenge durch den Konfigurationsparameter **cf_db_mem_sz** gesteuert.

Das Speicherlimit, das durch den Parameter **cf_db_mem_sz** auf Datenbankebene definiert wird, wird durch den Parameter **cf_mem_sz** auf Instanzebene beschränkt. Daher wird für den Parameter **cf_db_mem_sz** eine Obergrenze angegeben: Er darf diesen Speichergrenzwert nicht überschreiten, um dem Speicherbedarf für die CF-Datenbankstrukturen zu entsprechen. Der Gesamtsummenwert der Parameter **cf_db_mem_sz** für alle aktiven Datenbanken darf den Wert für **cf_mem_sz** für die Instanz nicht überschreiten.

Und obgleich der Parameter **cf_db_mem_sz** die Gesamtspeichermenge steuert, die von allen CF-Strukturen für eine Datenbank verwendet werden kann, kann der Datenbankmanager zu keinem Zeitpunkt einen Speicherblock reservieren, der dem Wert für **cf_db_mem_sz** entspricht.

Die Konfigurationsparameter, die die Speichermenge für die einzelnen CF-Strukturen für eine Datenbank steuern, lauten folgendermaßen:

- **cf_gbp_sz**
- **cf_lock_sz**
- **cf_sca_sz**

Die CF verwendet intern 3840 4-K-Seiten. Diese Seitenanzahl beruht auf der durch den Parameter **cf_mem_sz** festgelegten Anzahl. Zusätzlich werden für jede aktive Datenbank in der Instanz 256 4-KB-Seiten vom Parameterwert **cf_mem_sz** reserviert. Diese zusätzlichen Seiten sind nur enthalten, wenn der Parameter **cf_mem_sz** manuell festgelegt wird.

Automatische Konfiguration

Die Standardeinstellung für den Parameter **cf_mem_sz** ist AUTOMATIC. Dies bedeutet, dass die für die CF verfügbare Speichermenge unter Verwendung des auf dem CF-Host verfügbaren Gesamtspeichers berechnet wird. Der Parameter wird dynamisch auf einen der folgenden Werte gesetzt:

- Ein geeigneter Prozentsatz, in der Regel 70 % bis 90 % des verfügbaren Gesamtspeichers auf dem Host der CF.
- Der Umfang der freien Speichermenge auf dem Host der CF.

Auf welchen der beiden Werte der Parameter gesetzt wird, hängt davon ab, welcher Wert der niedrigere ist.

Die Standardeinstellung für den Parameter **cf_db_mem_sz** ist AUTOMATIC. Der tatsächliche Wert wird basierend auf den Werten der Parameter **DB2_DATABASE_CF_MEMORY**, **cf_mem_sz** und **numdb** bei der erstmaligen Aktivierung der Datenbank für ein Member bestimmt.

Ein anderer Faktor, der bei der automatischen Berechnung in Betracht gezogen wird, ist, ob die CF und irgendwelche DB2-Member koexistieren.

Online-Konfigurationen

Der Parameter **cf_mem_sz** kann nicht online konfiguriert werden. Der CF-Server muss neu gestartet werden, damit der neue Wert für **cf_mem_sz** wirksam wird.

Der Parameter **cf_db_mem_sz** kann zwar online konfiguriert werden, es müssen jedoch einige Einschränkungen beachtet werden:

- Der Wert für **cf_db_mem_sz** muss immer kleiner sein als der aktuelle Wert für **cf_mem_sz**. Wenn Sie den Parameter **cf_db_mem_sz** auf einen Wert erhöhen möchten, der höher ist als der aktuelle Wert für **cf_mem_sz**, müssen Sie zunächst den Wert für **cf_mem_sz** erhöhen.
- Selbst wenn der Parameter **cf_db_mem_sz** auf einen Wert erhöht wird, der kleiner ist als der Parameter **cf_mem_sz**, führen die folgenden Situationen zu einem Fehler:
  - Bei der Anforderung für eine Online-Erhöhung eines Speicherparameters für eine CF-Struktur, wobei die Summe der Speicherparameter für die CF-Struktur zwar den Wert für **cf_db_mem_sz** *nicht überschreitet*, jedoch den durch den Parameter **cf_mem_sz** festgelegten oberen Grenzwert *überschreitet*. Es ist nicht genügend Hauptspeicher in der CF vorhanden, um die Speicheranforderung für die CF-Struktur zu erfüllen, und es wird ein Fehler ausgegeben.
- Der Wert für **cf_db_mem_sz** muss immer größer sein als die Summe der Parameter **cf_gbp_sz**, **cf_lock_sz** und **cf_sca_sz**. Wenn der Wert für **cf_db_mem_sz** verringert wird, müssen die Werte dieser drei Parameter für die CF-Struktur ebenfalls verringert werden, um dem durch den Parameter **cf_db_mem_sz** festgelegten geänderten Speichergrenzwert zu entsprechen. Wird dies nicht vorgenommen, so wird ein Fehler ausgegeben, da der Parameter **cf_db_mem_sz** nicht mehr genügend Speicher zur Verfügung stellt.

Die Parameter **cf_gbp_sz**, **cf_lock_sz** und **cf_sca_sz** können zwar online konfiguriert werden, es müssen jedoch einige Einschränkungen beachtet werden:

- Jeglicher Versuch, diese Parameter zu ändern, kann zu einer Zeitüberschreitung führen, wenn das System ausgelastet ist oder zum Zeitpunkt des Sendens der Anforderung nicht genügend CF-Speicher verfügbar ist.

Anmerkung: Selbst wenn zum Zeitpunkt des Sendens der Anforderung nicht genügend Speicher zur Verfügung stehen sollte, versucht das System weiterhin, die Aktualisierung durchzuführen für den Fall, dass doch noch genügend Speicherplatz zum Erfüllen der Anforderung verfügbar wird. Die Aktualisierungsoperation wartet, bis die Verarbeitung abgeschlossen ist.

- Wenn während der Aktualisierung eines Speicherparameters für eine CF-Struktur eine zweite Anforderung zur Aktualisierung derselben Struktur initialisiert wird, wird ein Fehler ausgegeben.

Mehrere aktive Datenbanken

In einer DB2 pureScale-Umgebung mit mehreren aktiven Datenbanken müssen bei der Planung der Speicherkonfigurationsanforderungen für die einzelnen aktiven Datenbanken zusätzliche Aspekte berücksichtigt werden. Der wichtigste Konfigurationsparameter in diesem Kontext ist der Parameter **DB2_DATABASE_CF_MEMORY**. Dieser Parameter wird gemeinsam mit den anderen Konfigurationsparametern der CF verwendet, um den jeweiligen Prozentsatz des gesamten Speichers der CF anzugeben, der den einzelnen Datenbanken zugeordnet ist. Der Parameter **DB2_DATABASE_CF_MEMORY** wird als Datentyp 'float' ausgedrückt. Wenn er:

einen Wert von -1
aufweist, entspricht der dynamisch erstellte Wert des Parameters **cf_db_mem_sz** für die aktive Datenbank dem Ergebnis der folgenden Berechnung: **cf_mem_sz** geteilt durch **numdb**:

$$cf\_db\_mem\_sz = \frac{cf\_mem\_sz}{numdb}$$

Dies bedeutet, dass jede aktive Datenbank den gleichen Speicheranteil vom Parameter **cf_mem_sz** zugewiesen bekommt, wenn für die Datenbank der Parameter **cf_db_mem_sz** auf AUTOMATIC gesetzt ist.

Anmerkung: Wenn der berechnete Wert für **cf_db_mem_sz** kleiner ist als der Mindestwert des Parameters **cf_db_mem_sz**, wird der Wert für **cf_db_mem_sz** automatisch auf den zulässigen Mindestwert von **cf_db_mem_sz** gesetzt.

einen Wert von 0
aufweist, muss für alle Datenbanken in der DB2 pureScale-Umgebung der Wert für **cf_db_mem_sz** manuell gesetzt werden. Ist für **cf_db_mem_sz** hingegen der Wert AUTOMATIC festgelegt, wird der Datenbank der zulässige Mindestwert von 32.768 zugeordnet.

einen Wert zwischen 0 und 100
aufweist, entspricht der Wert für den Parameter **cf_db_mem_sz** dem Ergebnis der folgenden Berechnung:

$$cf\_db\_mem\_sz = cf\_mem\_sz * \frac{db2\_database\_cf\_memory}{100}$$

Anmerkung: Wenn der berechnete Wert für **cf_db_mem_sz** kleiner ist als der Mindestwert des Parameters **cf_db_mem_sz**, wird der Wert für **cf_db_mem_sz** automatisch auf den zulässigen Mindestwert von **cf_db_mem_sz** gesetzt.

einen Wert von 100
aufweist, kann nur eine Datenbank in der DB2 pureScale-Umgebung aktiv sein, die den gesamten durch den Parameter **cf_mem_sz** definierten CF-Speicher erhält.

In Konfigurationen, in denen mithilfe von **DB2_DATABASE_CF_MEMORY** der Wert für den Parameter **cf_db_mem_sz** dynamisch berechnet wird, darf der Wert für **cf_db_mem_sz** nie kleiner sein als der zulässige Mindestwert (unabhängig von der Einstellung des Parameters **DB2_DATABASE_CF_MEMORY**).

Einschränkungen

Eine automatische Leistungsoptimierung von CF-Speicher zwischen Datenbanken wird nicht unterstützt.

Beispiele

Beispiel 1: In diesem Beispiel wurde die Registrierdatenbankvariable **DB2_DATABASE_CF_MEMORY** nicht geändert und weist immer noch die Standardeinstellung „100" auf. Daher erhält nur eine Datenbank den gesamten CF-Speicher (eine Speichermenge, die dem durch den Parameter **cf_mem_sz** definierten Wert entspricht). Wenn eine zweite Datenbank erstellt oder gestartet wird, schlägt der Befehl mit einem Fehler über unzureichenden Speicher fehl. Um eine solche Situation zu vermeiden und eine zweite Datenbank mit gleichen Speicheranteilen erstellen zu können, müssen die CF-Speicherparameter entsprechend geändert werden (unter der Annahme, dass **cf_db_mem_sz** auf AUTOMATIC gesetzt ist):

```
db2stop
db2 update dbm cfg using numdb 2
db2set db2_database_cf_memory=50
db2 start
```

Beispiel 2: In diesem Beispiel wird die Registrierungsdatenbankvariable **DB2_DATABASE_CF_MEMORY** so geändert, dass allen Datenbanken die gleichen CF-Speicheranteile zugeordnet werden (unter der Annahme, dass zwei aktive Datenbanken vorhanden sind):

```
db2stop
db2 update dbm cfg using numdb 2
db2set db2_database_cf_memory=-1
db2 start
db2 connect to INVOICES      # success
db2 connect to HRDEPT        # success
```

Beispiel 3: In diesem Beispiel wird die Registrierdatenbankvariable **DB2_DATABASE_CF_MEMORY** so geändert, dass allen 200 Datenbanken die gleichen CF-Speicheranteile zugeordnet werden (unter der Annahme, dass 200 aktive Datenbanken vorhanden sind):

```
db2stop
db2 update dbm cfg using numdb 200
db2set db2_database_cf_memory=0.5
db2 start
```

Verhalten der Duplexunterstützung der Struktur mit einer sekundären Cluster-Caching-Funktion

Beim Betrieb in einer hoch verfügbaren Umgebung müssen Sie mehrere Cluster-Caching-Funktionen verwenden, um eine kontinuierliche Unterstützung bereitzustellen, wenn eine Cluster-Caching-Funktion (auch als CF bezeichnet) ausfällt.

Die folgenden Strukturen werden von der Cluster-Caching-Funktion unterstützt:

- Gruppenpufferpool (GBP)
- CF-Sperrenmanager (LOCK)
- Gemeinsamer Kommunikationsbereich (SCA)

Der primäre und der sekundäre CF-Server verwenden in einer Hochverfügbarkeitsumgebung eine identische Hostkonfiguration. Es wird empfohlen, dass die Hostspezifikationen wie CPU-Geschwindigkeit, Anzahl der CPUs und die Kapazität des Arbeitsspeichers (RAM) für beide CF-Server übereinstimmen. Die beiden CF-Server verwenden nur einen Satz von Konfigurationsparametern und greifen auf dieselben Portnummern zu.

Die CF besitzt konfigurierbare Strukturen: den Gruppenpufferpool (GBP), den Sperrenmanager (LOCK) und den gemeinsamer Kommunikationsbereich (SCA, Shared Communication Area). Alle Strukturen unterstützen Onlineaktualisierungen. Wenn der für diese Strukturen zugeordnete Speicher vergrößert wird, (Strukturvergrößerung), wird die Anforderung zunächst an den sekundären CF-Server und anschließend an den primären CF-Server gesendet. Die umgekehrte Reihenfolge gilt für das Verkleinern des für diese Strukturen zugeordneten Speichers (Strukturverkleinerung). Die Verkleinerungsanforderung wird zunächst an den primären CF-Server und anschließend an den sekundären CF-Server gesendet. Eine Struktur in der sekundären CF ist immer mindestens so groß wie die Struktur in der primären CF oder größer, um zu gewährleisten, dass nach einer CF-Funktionsübernahme keine Speicherprobleme auftreten.

Wenn entweder die primäre CF oder die sekundäre CF einen CF-Konfigurationsparameter nicht aktualisieren kann, führt die andere CF einen Rollback auf die vorherige Einstellung für den betreffenden CF-Konfigurationsparameter durch. Dieser Rollback stellt sicher, dass ausreichend Strukturspeicher auf beiden CFs verfügbar ist.

Standardmäßig wartet eine Anforderung zur Größenänderung mit der Rückkehr zum Client ab, bis die Größenänderungsoperation abgeschlossen ist. Wenn Sie die Größenänderungsanforderung durch eine Benutzersteuerungsaktion (z. B. durch die Tastenkombination **Strg+C**) unterbrechen, wird die Größenänderungsoperation auf beiden CFs abgebrochen. Die Größe der Struktur auf der sekundären CF wird in diesem Fall angepasst, sodass sie mit dem Wert auf der primären CF übereinstimmt. Sie können eine Größenänderungsanforderung mithilfe des Befehls `DB2 GET DB CFG SHOW DETAIL` überwachen.

Konfigurationsparameter des Aufnahmedienstprogramms

commit_count - Anzahl Commits (Konfigurationsparameter)

Dieser Parameter gibt an, wie viele Zeilen jeder Flusher in einer einzelnen Transaktion schreibt, bevor eine Commitoperation durchgeführt wird.

Konfigurationstyp
Aufnahmedienstprogramm

Gilt für
Aufnahmedienstprogramm

Parametertyp
Konfigurierbar

Standardwert [Bereich]
0 [0 bis max. 32 Bit, ganze Zahl mit Vorzeichen]

Wenn der Wert für **`commit_count`** kein Vielfaches von 1.000 ist, wird auf das nächste Vielfache gerundet und das Aufnahmedienstprogramm gibt eine Warnung (SQL2903W) aus.

Wenn **commit_count** auf 0 (Standardwert) gesetzt ist, wird **commit_period** verwendet (d. h. das Aufnahmedienstprogramm schreibt Transaktionen allein aufgrund der abgelaufenen Zeit fest. Wenn Transaktionen allein aufgrund der Zeilenanzahl festgeschrieben werden sollen, müssen Sie **commit_count** auf einen Wert ungleich Null und **commit_period** auf 0 setzen.

Wenn weder **commit_count** noch **commit_period** angegeben ist, wird für **commit_period** die Standardeinstellung (1 Sekunde) verwendet.

Wenn sowohl **commit_count** als auch **commit_period** angegeben ist, berücksichtigt das Aufnahmedienstprogramm beide Werte, d. h. eine Commitoperation wird ausgegeben, nachdem die angegebene Anzahl Zeilen geschrieben wurde, oder wenn innerhalb der angegebenen Anzahl Sekunden keine Commitoperation stattgefunden hat.

Empfehlungen

Verwenden Sie bei kleiner Zeilengröße einen großen Wert für **commit_count**. Geben Sie bei großer Zeilengröße einen kleinen Wert für **commit_count** an.

Wenn keine anderen Anwendungen aktiv sind, kann der absolute Maximalwert für die Anzahl Commits mit der folgenden Formel grob geschätzt werden:

```
(logfilsiz * (logprimary + logsecond) * 4KB) dividiert durch (geschätzte Zeilengröße + Systemaufwand)
   dividiert durch (Gesamtzahl Flusher)
```

Wenn andere Anwendungen ausgeführt werden, ist der Maximalwert für die Anzahl Commits niedriger.

Wenn **commit_count** auf einen zu hohen Wert gesetzt wird, steigt das Risiko eines Überlaufs der Sperrenliste oder des Transaktionsprotokolls. In diesem Fall wird der Befehl INGEST mit einem Fehler SQL0912N oder SQL0964C beendet. Wenn SQL0912N oder SQL0964C ausgegeben wird und der Konfigurationsparameter **retry_count** gesetzt ist, führt das Aufnahmedienstprogramm die beiden folgenden Aktionen aus:

- Einstellung von **commit_count** und/oder **commit_period** anpassen und Warnung ausgeben
- Commit ausgeben und Operation wiederholen

commit_period - Commit-Zeitraum (Konfigurationsparameter)

Gibt die Anzahl Sekunden zwischen festgeschriebenen Transaktionen an.

Konfigurationstyp
Aufnahmedienstprogramm

Gilt für
Aufnahmedienstprogramm

Parametertyp
Konfigurierbar

Standardwert [Bereich]
1 [0 bis 2.678.400 (31 Tage)]

Maßeinheit
Sekunden

Bei jeder Flushoperation prüft das Aufnahmedienstprogramm, wie viele Sekunden seit der letzten Commitoperation vergangen sind. Wenn dieser Wert größer oder gleich der Einstellung für **commit_period** ist, führt das Aufnahmedienstprogramm eine Commitoperation aus.

Wenn **commit_period** auf 0 gesetzt ist, wird der Konfigurationsparameter **commit_count** verwendet, d. h. Transaktionen sind auf der Basis der Zeilenanzahl festzuschreiben.

Wenn weder **commit_count** noch **commit_period** angegeben ist, wird für **commit_period** die Standardeinstellung (1 Sekunde) verwendet.

Wenn sowohl **commit_count** als auch **commit_period** angegeben ist, berücksichtigt das Aufnahmedienstprogramm beide Werte, d. h. eine Commitoperation wird ausgegeben, nachdem die angegebene Anzahl Zeilen geschrieben wurde oder wenn innerhalb der angegebenen Anzahl Sekunden keine Commitoperation stattgefunden hat.

Empfehlungen

Geben Sie bei kleiner Zeilengröße einen großen Wert für **commit_period** an. Verwenden Sie bei großer Zeilengröße einen kleinen Wert für **commit_period**.

Wenn **commit_period** auf einen zu hohen Wert gesetzt wird, steigt das Risiko eines Überlaufs der Sperrenliste oder des Transaktionsprotokolls. In diesem Fall wird der Befehl INGEST mit einem Fehler SQL0912N oder SQL0964C beendet. Wenn SQL0912N oder SQL0964C ausgegeben wird und der Konfigurationsparameter **retry_count** gesetzt ist, führt das Aufnahmedienstprogramm die beiden folgenden Aktionen aus:

- Einstellung von **commit_count** und/oder **commit_period** anpassen und Warnung ausgeben
- Commit ausgeben und Operation wiederholen

num_flushers_per_partition - Anzahl der Flusher pro Datenbankpartition (Konfigurationsparameter)

Gibt die Anzahl der Flusher an, die für jede Datenbankpartition zugeordnet werden können.

Konfigurationstyp

Aufnahmedienstprogramm

Gilt für

Aufnahmedienstprogramm

Parametertyp

Konfigurierbar

Standardwert [Bereich]

max(1, ((Anzahl logische CPUs)/2)/(Anzahl Partitionen)) [0 bis Systemressourcen]

Wenn **num_flushers_per_partition** auf 0 gesetzt ist, wird ein Flusher für alle Partitionen verwendet.

Anmerkung: Für die Operationen DELETE, MERGE oder UPDATE reduziert das Dienstprogramm diesen Parameter in manchen Fällen auf den Wert 0 oder 1, um das Risiko für das Auftreten von Deadlocks (SQL2903W) zu verringern.

num_formatters - Anzahl der Formatierungsprogramme (Konfigurationsparameter)

Gibt an, wie viele Formatierungsprogramme zugeordnet werden sollen.

Konfigurationstyp
Aufnahmedienstprogramm

Gilt für
Aufnahmedienstprogramm

Parametertyp
Konfigurierbar

Standardwert [Bereich]
max(1, ((Anzahl logische CPUs)/2)) [1 bis Systemressourcen]

Im Allgemeinen ist der Standardwert die optimale Einstellung. Wenn Sie jedoch den Parameter `DUMPFILE` (oder `BADFILE`) im Befehl **`INGEST`** angeben und das Aufnahmedienstprogramm die fehlerhaften Datensätze in derselben Reihenfolge schreiben soll, in der sie in der Eingabedatei vorkommen, müssen Sie den Parameter **`num_formatters`** auf 1 setzen.

pipe_timeout - Pipe-Zeitlimit (Konfigurationsparameter)

Dieser Parameter gibt an, wie viele Sekunden maximal auf Daten gewartet werden soll, wenn die Eingabequelle eine Pipe ist.

Konfigurationstyp
Aufnahmedienstprogramm

Gilt für
Aufnahmedienstprogramm

Parametertyp
Konfigurierbar

Standardwert [Bereich]
600 Sekunden (10 Minuten) [0 bis 2.678.400 (31 Tage)]

Maßeinheit
Sekunden

Wenn **`pipe_timeout`** auf 0 gesetzt ist, wartet das Aufnahmedienstprogramm unbegrenzt lange. Treffen im angegebenen Zeitraum keine Daten ein, wird der Befehl **`INGEST`** beendet und ein Fehler zurückgegeben.

retry_count - Wiederholungszähler (Konfigurationsparameter)

Gibt die Anzahl der Wiederholungen für eine fehlgeschlagene aber wiederherstellbare Transaktion an.

Konfigurationstyp
Aufnahmedienstprogramm

Gilt für
Aufnahmedienstprogramm

Parametertyp
Konfigurierbar

Standardwert [Bereich]
0 [0 bis 1.000]

Das Aufnahmedienstprogramm wiederholt nur Transaktionen, die aus einem der folgenden Gründe fehlgeschlagen sind:

- Eine Verbindung ist fehlgeschlagen und konnte wiederhergestellt werden.
- Ein Deadlock oder eine Zeitlimitüberschreitung mit automatischem Rollback ist aufgetreten.
- Aufgrund eines Systemfehlers wurde die Arbeitseinheit rückgängig gemacht.
- Virtueller Speicher oder eine Datenbankressource ist nicht verfügbar.

retry_period - Wiederholungszeitlimit (Konfigurationsparameter)

Gibt die Wartezeit in Sekunden bis zur Wiederholung einer fehlgeschlagenen aber wiederherstellbaren Transaktion an.

Konfigurationstyp
Aufnahmedienstprogramm

Gilt für
Aufnahmedienstprogramm

Parametertyp
Konfigurierbar

Standardwert [Bereich]
0 [0 bis 2.678.400 (31 Tage)]

Maßeinheit
Sekunden

Das Aufnahmedienstprogramm wiederholt nur Transaktionen, die aus einem der folgenden Gründe fehlgeschlagen sind:

- Eine Verbindung ist fehlgeschlagen und konnte wiederhergestellt werden.
- Ein Deadlock oder eine Zeitlimitüberschreitung mit automatischem Rollback ist aufgetreten.
- Aufgrund eines Systemfehlers wurde die Arbeitseinheit rückgängig gemacht.
- Virtueller Speicher oder eine Datenbankressource ist nicht verfügbar.

shm_max_size - Maximale Größe des gemeinsam genutzten Speichers (Konfigurationsparameter)

Gibt die maximale Größe des gemeinsam genutzten IPC-Speichers in Byte an (IPC = Inter Process Communication, Interprozesskommunikation). Weil das Aufnahmedienstprogramm auf dem Client ausgeführt wird, wird dieser Speicher auf der Clientmaschine zugeordnet.

Konfigurationstyp
Befehl 'Ingest'

Gilt für
Aufnahmedienstprogramm

Parametertyp
Konfigurierbar

Standardwert [Bereich]
1 GB (1.073.741.824) [*n* bis verfügbarer Speicher]

n wird wie folgt berechnet:

```
11000 +
(anzahl_transporter x 500) +
(NUM_FORMATTERS x 500) +
(anzahl_verwendete_partitionen x 50) +
(gesamtzahl_flusher x 4000) +
(MSG_BUF_COUNT x (100 + MSG_BUF_SIZE)) +
(anzahl_felder x 66300) +
(1.5 x NUM_FORMATTERS x summe_aller_feldlängen)
```

Dabei gilt Folgendes:

- *anzahl_transporter* ist die Anzahl der Eingabequellen (wenn die Operation INSERT oder REPLACE ist) oder andernfalls 1.
- *anzahl_verwendete_partitionen* ist die Anzahl der Datenbankpartitionen, die von der Zieltabelle verwendet werden.
- *gesamtzahl_flusher* ist **num_flushers_per_partition** x *anzahl_verwendete_partitionen*.
- *summe_aller_feldlängen* ist die Gesamtzahl der Byte in allen Felddefinitionen.
- *anzahl_felder* ist die Anzahl der Felddefinitionen.

Maßeinheit
Byte

Teil 5. Anhänge und Schlussteil

Anhang A. Übersicht über technische Informationen zu DB2

Technische Informationen zu DB2 liegen in verschiedenen Formaten vor, die auf unterschiedliche Weise abgerufen werden können.

Die technischen Informationen zu DB2 stehen über die folgenden Tools und Methoden zur Verfügung:

- DB2 Information Center
  - Themen (zu Tasks, Konzepten und Referenzinformationen)
  - Beispielprogramme
  - Lernprogramme
- DB2-Bücher
  - PDF-Dateien (für den Download verfügbar)
  - PDF-Dateien (auf der DB2-PDF-DVD)
  - Gedruckte Bücher
- Hilfe für Befehlszeile
  - Hilfe für Befehle
  - Hilfe für Nachrichten

Anmerkung: Die Themen des DB2 Information Center werden häufiger aktualisiert als die PDF- und Hardcopybücher. Um stets die neuesten Informationen zur Verfügung zu haben, sollten Sie die Dokumentationsaktualisierungen installieren, sobald diese verfügbar sind, oder das DB2 Information Center unter ibm.com aufrufen.

Darüber hinaus können Sie auf zusätzliche technische Informationen zu DB2, wie beispielsweise technische Hinweise (Technotes), White Papers und IBM Redbooks, online über ibm.com zugreifen. Rufen Sie dazu die Website 'DB2 Information Management - Software - Library' unter http://www.ibm.com/software/data/sw-library/ auf.

Feedback zur Dokumentation

Senden Sie uns Ihr Feedback zur DB2-Dokumentation! Wenn Sie Anregungen zur Verbesserung der DB2-Dokumentation haben, senden Sie eine E-Mail an db2docs@ca.ibm.com. Das DB2-Dokumentationsteam bearbeitet das gesamte Feedback, kann jedoch nicht im Einzelnen auf Ihre E-Mails antworten. Nennen Sie uns, wenn möglich, konkrete Beispiele, sodass wir die Problemstellung besser beurteilen können. Wenn Sie uns Feedback zu einem bestimmten Thema oder einer bestimmten Hilfedatei senden, geben Sie den entsprechenden Titel sowie die URL an.

Verwenden Sie diese E-Mail-Adresse nicht, wenn Sie sich an den DB2-Kundendienst wenden möchten. Wenn ein technisches Problem bei DB2 vorliegt, das Sie mithilfe der Dokumentation nicht beheben können, fordern Sie beim zuständigen IBM Service-Center Unterstützung an.

Bibliothek mit technischen Informationen zu DB2 im Hardcopy- oder PDF-Format

Die folgenden Tabellen enthalten eine Beschreibung der DB2-Bibliothek, die im IBM Publications Center unter www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss zur Verfügung steht. Über die folgende Adresse können Sie englische Handbücher im PDF-Format sowie übersetzte Versionen zu DB2 Version 10.1 herunterladen: www.ibm.com/support/docview.wss?rs=71&uid=swg2700947.

In den Tabellen sind die Bücher, die in gedruckter Form zur Verfügung stehen, gekennzeichnet; möglicherweise sind diese in Ihrem Land oder Ihrer Region jedoch nicht verfügbar.

Die Formnummer wird bei jeder Aktualisierung eines Handbuchs erhöht. Anhand der nachfolgenden Liste können Sie sicherstellen, dass Sie die jeweils neueste Version des Handbuchs lesen.

Anmerkung: Das *DB2 Information Center* wird häufiger aktualisiert als die PDF- und Hardcopybücher.

Tabelle 141. Technische Informationen zu DB2

Name	IBM Form	In gedruckter Form verfügbar	Letzte Aktualisierung
Administrative API Reference	SC27-3864-00	Ja	April 2012
Administrative Routines and Views	SC27-3865-00	Nein	April 2012
Call Level Interface Guide and Reference Volume 1	SC27-3866-00	Ja	April 2012
Call Level Interface Guide and Reference Volume 2	SC27-3867-00	Ja	April 2012
Command Reference	SC27-3868-00	Ja	April 2012
Datenbankverwaltung - Konzepte und Konfiguration - Referenzinformationen	SC12-4673-00	Ja	April 2012
Dienstprogramme für das Versetzen von Daten - Handbuch und Referenz	SC12-4691-00	Ja	April 2012
Datenbanküberwachung - Handbuch und Referenz	SC12-4674-00	Ja	April 2012
Datenrecovery und hohe Verfügbarkeit - Handbuch und Referenz	SC12-4692-00	Ja	April 2012
Datenbanksicherheit	SC12-4693-00	Ja	April 2012
DB2 Workload Management - Handbuch und Referenz	SC12-4683-00	Ja	April 2012

Tabelle 141. Technische Informationen zu DB2 (Forts.)

Name	IBM Form	In gedruckter Form verfügbar	Letzte Aktualisierung
Developing ADO.NET and OLE DB Applications	SC27-3873-00	Ja	April 2012
Developing Embedded SQL Applications	SC27-3874-00	Ja	April 2012
Developing Java Applications	SC27-3875-00	Ja	April 2012
Developing Perl, PHP, Python, and Ruby on Rails Applications	SC27-3876-00	Nein	April 2012
Developing User-defined Routines (SQL and External)	SC27-3877-00	Ja	April 2012
Getting Started with Database Application Development	GI13-2046-00	Ja	April 2012
Installation und Verwaltung von DB2 unter Linux und Windows - Erste Schritte	GI11-3285-00	Ja	April 2012
Globalisierung	SC12-4694-00	Ja	April 2012
DB2-Server - Installation	SC12-4677-00	Ja	April 2012
IBM Data Server-Clients - Installation	SC12-4678-00	Nein	April 2012
Fehlernachrichten, Band 1	SC12-4686-00	Nein	April 2012
Fehlernachrichten, Band 2	SC12-4687-00	Nein	April 2012
Net Search Extender - Verwaltung und Benutzerhandbuch	SC12-4689-00	Nein	April 2012
Partitionierung und Clustering	SC12-4695-00	Ja	April 2012
pureXML - Handbuch	SC12-4684-00	Ja	April 2012
Spatial Extender - Benutzer- und Referenzhandbuch	SC12-4688-00	Nein	April 2012
SQL Procedural Languages: Application Enablement and Support	SC27-3896-00	Ja	April 2012
SQL Reference Volume 1	SC27-3885-00	Ja	April 2012
SQL Reference Volume 2	SC27-3886-00	Ja	April 2012
Text Search	SC12-4674-00	Ja	April 2012
Fehlerbehebung und Optimieren der Datenbankleistung	SC12-4675-00	Ja	April 2012

Tabelle 141. Technische Informationen zu DB2 (Forts.)

Name	IBM Form	In gedruckter Form verfügbar	Letzte Aktualisierung
Upgrade auf DB2 Version 10.1	SC12-4676-00	Ja	April 2012
Neuerungen in DB2 Version 10.1	SC12-4682-00	Ja	April 2012
XQuery - Referenz	SC12-4685-00	Nein	April 2012

Tabelle 142. Technische Informationen zu DB2 Connect

Name	IBM Form	In gedruckter Form verfügbar	Letzte Aktualisierung
DB2 Connect - Installation und Konfiguration von DB2 Connect Personal Edition	SC12-4679-00	Ja	April 2012
DB2 Connect - Installation und Konfiguration von DB2 Connect-Servern	SC12-4680-00	Ja	April 2012
DB2 Connect - Benutzerhandbuch	SC12-4681-00	Ja	April 2012

Aufrufen der Hilfe für den SQL-Status über den Befehlszeilenprozessor

DB2-Produkte geben für Bedingungen, die aufgrund einer SQL-Anweisung generiert werden können, einen SQLSTATE-Wert zurück. Die SQLSTATE-Hilfe erläutert die Bedeutung der SQL-Statuswerte und der SQL-Statusklassencodes.

Vorgehensweise

Zum Starten der Hilfe für SQL-Statuswerte müssen Sie den Befehlszeilenprozessor öffnen und Folgendes eingeben:

```
? SQL-Status oder ? Klassencode
```

Hierbei steht *SQL-Status* für einen gültigen fünfstelligen SQL-Statuswert und *Klassencode* für die ersten beiden Ziffern dieses Statuswerts.
So kann beispielsweise durch die Eingabe von `? 08003` Hilfe für den SQL-Statuswert 08003 angezeigt werden, durch die Eingabe von `? 08` Hilfe für den Klassencode 08.

Zugriff auf verschiedene Versionen des DB2 Information Center

Die Dokumentation für andere Versionen der DB2-Produkte finden Sie in den jeweiligen Information Centers unter ibm.com.

Informationen zu diesem Vorgang

Für Themen aus DB2 Version 10.1 lautet die URL für das *DB2 Information Center* http://publib.boulder.ibm.com/infocenter/db2luw/v10r1.

Für Themen aus DB2 Version 9.8 lautet die URL des *DB2 Information Center* http://publib.boulder.ibm.com/infocenter/db2luw/v9r8/.

Für Themen aus DB2 Version 9.7 lautet die URL des *DB2 Information Center* http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/.

Für Themen aus DB2 Version 9.5 lautet die URL des *DB2 Information Center* http://publib.boulder.ibm.com/infocenter/db2luw/v9r5.

Für Themen aus DB2 Version 9.1 lautet die URL des *DB2 Information Center* http://publib.boulder.ibm.com/infocenter/db2luw/v9/.

Für Themen aus DB2 Version 8 lautet die URL des *DB2 Information Center* http://publib.boulder.ibm.com/infocenter/db2luw/v8/.

Aktualisieren des auf Ihrem Computer oder Intranet-Server installierten DB2 Information Center

Ein lokal installiertes DB2 Information Center muss regelmäßig aktualisiert werden.

Vorbereitende Schritte

Ein DB2 Version 10.1 Information Center muss bereits installiert sein. Einzelheiten hierzu finden Sie unter „Installation des DB2 Information Center mit dem DB2-Installationsassistenten" in *DB2-Server - Installation*. Alle für die Installation des Information Center geltenden Voraussetzungen und Einschränkungen gelten auch für die Aktualisierung des Information Center.

Informationen zu diesem Vorgang

Ein vorhandenes DB2 Information Center kann automatisch oder manuell aktualisiert werden:

- Mit automatischen Aktualisierungen werden vorhandene Komponenten und Sprachen des Information Center aktualisiert. Ein Vorteil von automatischen Aktualisierungen ist, dass das Information Center im Vergleich zu einer manuellen Aktualisierung nur für einen kurzenZeitraum nicht verfügbar ist. Darüber hinaus können automatische Aktualisierungen so konfiguriert werden, dass sie als Teil anderer, regelmäßig ausgeführter Stapeljobs ausgeführt werden.
- Mit manuellen Aktualisierungen können Sie vorhandene Komponenten und Sprachen des Information Center aktualisieren. Automatische Aktualisierungen reduzieren die Ausfallzeiten während des Aktualisierungsprozesses, Sie müssen jedoch den manuellen Prozess verwenden, wenn Sie Komponenten oder Sprachen hinzufügen möchten. Beispiel: Ein lokales Information Center wurde ursprünglich sowohl mit englischer als auch mit französischer Sprachunterstützung installiert; nun soll auch die deutsche Sprachunterstützung installiert werden. Bei einer manuellen Aktualisierung werden sowohl eine Installation der deutschen Sprachunterstützung als auch eine Aktualisierung der vorhandenen Komponenten und Sprachen des Information Center durchgeführt. Sie müssen jedoch bei einer manuellen Aktualisierung das Information Center manuell stoppen, aktualisieren und erneut starten. Das Information Center ist während des gesamten Aktualisierungsprozesses nicht verfügbar. Während des automatischen Aktualisierungsprozesses kommt es zu einem Ausfall des Information Center, und es wird erst wieder nach der Aktualisierung erneut gestartet.

Dieser Abschnitt enthält Details zum Prozess der automatischen Aktualisierung. Anweisungen zur manuellen Aktualisierung finden Sie im Abschnitt „Manuelles Aktualisieren des auf Ihrem Computer oder Intranet-Server installierten DB2 Information Center".

Vorgehensweise

Gehen Sie wie folgt vor, um das auf Ihrem Computer bzw. Intranet-Server installierte DB2 Information Center automatisch zu aktualisieren:

1. Unter Linux:
   a. Navigieren Sie zu dem Pfad, in dem das Information Center installiert ist. Standardmäßig ist das DB2 Information Center im Verzeichnis `/opt/ibm/db2ic/V10.1` installiert.
   b. Navigieren Sie vom Installationsverzeichnis in das Verzeichnis `doc/bin`.
   c. Führen Sie das Script `update-ic` aus:
      ```
      update-ic
      ```
2. Unter Windows:
   a. Öffnen Sie ein Befehlsfenster.
   b. Navigieren Sie zu dem Pfad, in dem das Information Center installiert ist. Standardmäßig ist das DB2 Information Center im Verzeichnis `<Programme>\IBM\DB2 Information Center\Version 10.1` installiert, wobei `<Programme>` das Verzeichnis der Programmdateien angibt.
   c. Navigieren Sie vom Installationsverzeichnis in das Verzeichnis `doc\bin`.
   d. Führen Sie die Datei `update-ic.bat` aus:
      ```
      update-ic.bat
      ```

Ergebnisse

Das DB2 Information Center wird automatisch erneut gestartet. Standen Aktualisierungen zur Verfügung, zeigt das Information Center die neuen und aktualisierten Abschnitte an. Waren keine Aktualisierungen für das Information Center verfügbar, wird eine entsprechende Nachricht zum Protokoll hinzugefügt. Die Protokolldatei befindet sich im Verzeichnis `doc\eclipse\configuration`. Der Name der Protokolldatei ist eine Zufallszahl. Beispiel: `1239053440785.log`.

Manuelles Aktualisieren des auf Ihrem Computer oder Intranet-Server installierten DB2 Information Center

Wenn Sie das DB2 Information Center lokal installiert haben, können Sie Dokumentationsaktualisierungen von IBM abrufen und installieren.

Informationen zu diesem Vorgang

Zur manuellen Aktualisierung des lokal installierten *DB2 Information Center* sind die folgenden Schritte erforderlich:

1. Stoppen Sie das *DB2 Information Center* auf Ihrem Computer und starten Sie das Information Center im Standalone-Modus erneut. Die Ausführung des Information Center im Standalone-Modus verhindert, dass andere Benutzer in Ihrem Netz auf das Information Center zugreifen, und ermöglicht das Anwenden von Aktualisierungen. Die Workstationversion des DB2 Information Center wird stets im Standalone-Modus ausgeführt.

2. Verwenden Sie die Aktualisierungsfunktion, um zu prüfen, welche Aktualisierungen verfügbar sind. Falls Aktualisierungen verfügbar sind, die Sie installieren müssen, können Sie die Aktualisierungsfunktion verwenden, um diese abzurufen und zu installieren.

   Anmerkung: Wenn es in der verwendeten Umgebung erforderlich ist, die Aktualisierungen für das *DB2 Information Center* auf einer Maschine zu installieren, die nicht über ein Verbindung zum Internet verfügt, spiegeln Sie die Aktualisierungssite auf ein lokales Dateisystem und verwenden Sie dabei eine Maschine, die mit dem Internet verbunden ist und auf der das *DB2 Information Center* installiert ist. Wenn viele Benutzer Ihres Netzes die Dokumentationsaktualisierungen installieren sollen, können Sie die Zeit, die jeder einzelne Benutzer für die Aktualisierungen benötigt, reduzieren, indem Sie die Aktualisierungssite lokal spiegeln und ein Proxy dafür erstellen.
   Ist dies der Fall, verwenden Sie die Aktualisierungsfunktion, um die Pakete abzurufen. Die Aktualisierungsfunktion ist jedoch nur im Standalone-Modus verfügbar.
3. Stoppen Sie das im Standalone-Modus gestartete Information Center und starten Sie das *DB2 Information Center* auf Ihrem Computer erneut.

Anmerkung: Unter Windows 2008 und Windows Vista (und neueren Versionen) müssen die in diesem Abschnitt aufgeführten Befehle mit Administratorberechtigung ausgeführt werden. Zum Öffnen einer Eingabeaufforderung oder eines Grafiktools mit vollen Administratorberechtigungen klicken Sie mit der rechten Maustaste die Verknüpfung an und wählen Sie **Als Administrator ausführen** aus.

Vorgehensweise

Gehen Sie wie folgt vor, um das auf Ihrem Computer bzw. Intranet-Server installierte *DB2 Information Center* zu aktualisieren:

1. Stoppen Sie das *DB2 Information Center*.
   - Unter Windows: Klicken Sie **Start** > **Systemsteuerung** > **Verwaltung** > **Dienste** an. Klicken Sie mit der rechten Maustaste das **DB2 Information Center** an und wählen Sie **Beenden** aus.
   - Unter Linux: Geben Sie den folgenden Befehl ein:
     ```
     /etc/init.d/db2icdv10 stop
     ```
2. Starten Sie das Information Center im Standalone-Modus.
   - Unter Windows:
     a. Öffnen Sie ein Befehlsfenster.
     b. Navigieren Sie zu dem Pfad, in dem das Information Center installiert ist. Standardmäßig ist das *DB2 Information Center* im Verzeichnis *Programme*\IBM\DB2 Information Center\Version 10.1 installiert, wobei *Programme* das Verzeichnis der Programmdateien angibt.
     c. Navigieren Sie vom Installationsverzeichnis in das Verzeichnis doc\bin.
     d. Führen Sie die Datei help_start.bat aus:
        ```
        help_start.bat
        ```
   - Unter Linux:
     a. Navigieren Sie zu dem Pfad, in dem das Information Center installiert ist. Standardmäßig ist das *DB2 Information Center* im Verzeichnis /opt/ibm/db2ic/V10.1 installiert.
     b. Navigieren Sie vom Installationsverzeichnis in das Verzeichnis doc/bin.
     c. Führen Sie das Script help_start aus:

```
help_start
```

Der standardmäßig auf dem System verwendete Web-Browser wird geöffnet und zeigt die Standalone-Version des Information Center an.

3. Klicken Sie die Aktualisierungsschaltfläche () an. (JavaScript muss im verwendeten Browser aktiviert sein.) Klicken Sie im rechten Fenster des Information Center die Schaltfläche für die Suche nach Aktualisierungen an. Eine Liste der Aktualisierungen für die vorhandene Dokumentation wird angezeigt.
4. Wählen Sie zum Initiieren des Installationsprozesses die gewünschten Aktualisierungen aus und klicken Sie anschließend die Schaltfläche für die Installation der Aktualisierungen an.
5. Klicken Sie nach Abschluss des Installationsprozesses **Fertigstellen** an.
6. Stoppen Sie das im Standalone-Modus gestartete Information Center:
   - Unter Windows: Navigieren Sie innerhalb des Installationsverzeichnisses zum Verzeichnis `doc\bin`, und führen Sie die Datei `help_end.bat` aus:
     ```
     help_end.bat
     ```
     Anmerkung: Die Stapeldatei `help_end` enthält die Befehle, die erforderlich sind, um die Prozesse, die mit der Stapeldatei `help_start` gestartet wurden, ordnungsgemäß zu stoppen. Verwenden Sie nicht die Tastenkombination `Strg+C` oder eine andere Methode, um `help_start.bat` zu stoppen.
   - Unter Linux: Navigieren Sie innerhalb des Installationsverzeichnisses zum Verzeichnis `doc/bin`, und führen Sie das Script `help_end` aus:
     ```
     help_end
     ```
     Anmerkung: Das Script `help_end` enthält die Befehle, die erforderlich sind, um die Prozesse, die mit dem Script `help_start` gestartet wurden, ordnungsgemäß zu stoppen. Verwenden Sie keine andere Methode, um das Script `help_start` zu stoppen.
7. Starten Sie das *DB2 Information Center* erneut.
   - Unter Windows: Klicken Sie **Start** > **Systemsteuerung** > **Verwaltung** > **Dienste** an. Klicken Sie mit der rechten Maustaste das **DB2 Information Center** an und wählen Sie **Start** aus.
   - Unter Linux: Geben Sie den folgenden Befehl ein:
     ```
     /etc/init.d/db2icdv10 start
     ```

Ergebnisse

Im aktualisierten *DB2 Information Center* werden die neuen und aktualisierten Themen angezeigt.

DB2-Lernprogramme

Die DB2-Lernprogramme unterstützen Sie dabei, sich mit den unterschiedlichen Aspekten der DB2-Produkte vertraut zu machen. Die Lerneinheiten bieten eine in einzelne Schritte unterteilte Anleitung.

Vorbereitungen

Die XHTML-Version des Lernprogramms kann über das Information Center unter http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/ angezeigt werden.

In einigen der Lerneinheiten werden Beispieldaten und Codebeispiele verwendet. Informationen zu bestimmten Voraussetzungen für die Ausführung der Tasks finden Sie in der Beschreibung des Lernprogramms.

DB2-Lernprogramme

Klicken Sie zum Anzeigen des Lernprogramms den Titel an.

„pureXML" in *pureXML - Handbuch*
Einrichten einer DB2-Datenbank, um XML-Daten zu speichern und Basisoperationen mit dem nativen XML-Datenspeicher auszuführen.

Informationen zur Fehlerbehebung in DB2

Es steht eine breite Palette verschiedener Informationen zur Fehlerbestimmung und Fehlerbehebung zur Verfügung, um Sie bei der Verwendung von DB2-Datenbankprodukten zu unterstützen.

DB2-Dokumentation
Informationen zur Fehlerbehebung stehen im Handbuch *Fehlerbehebung und Optimieren der Datenbankleistung* oder im Abschnitt mit grundlegenden Informationen zu Datenbanken im *DB2 Information Center* zur Verfügung, darunter:

- Informationen zum Eingrenzen und Aufdecken von Problemen mithilfe der Diagnosetools und -dienstprogramme von DB2.
- Lösungsvorschläge zu den am häufigsten auftretenden Problemen.
- Ratschläge zum Lösen anderer Probleme, die bei Verwendung der DB2-Datenbankprodukte auftreten können.

IBM Support Portal
Im IBM Support Portal finden Sie Informationen zu Problemen und den möglichen Ursachen und Fehlerbehebungsmaßnahmen. Die Website mit technischer Unterstützung enthält Links zu den neuesten DB2-Veröffentlichungen, technischen Hinweisen (TechNotes), APARs (Authorized Program Analysis Reports) und Fehlerkorrekturen, Fixpacks sowie weiteren Ressourcen. Sie können diese Wissensbasis nach möglichen Lösungen für aufgetretene Probleme durchsuchen.

Sie können auf das IBM Support Portal über die folgende Website zugreifen: http://www.ibm.com/support/entry/portal/Overview/Software/Information_Management/DB2_for_Linux,_UNIX_and_Windows.

Bedingungen

Die Berechtigungen zur Nutzung dieser Veröffentlichungen werden Ihnen auf der Basis der folgenden Bedingungen gewährt.

Anwendbarkeit: Diese Bedingungen gelten zusätzlich zu den Nutzungsbedingungen für die IBM Website.

Persönliche Nutzung: Sie dürfen diese Veröffentlichungen für Ihre persönliche, nicht kommerzielle Nutzung unter der Voraussetzung vervielfältigen, dass alle Eigentumsvermerke erhalten bleiben. Sie dürfen diese Veröffentlichungen oder Teile dieser Veröffentlichungen ohne ausdrückliche Genehmigung von IBM nicht weitergeben, anzeigen oder abgeleitete Werke davon erstellen.

Kommerzielle Nutzung: Sie dürfen diese Veröffentlichungen nur innerhalb Ihres Unternehmens und unter der Voraussetzung, dass alle Eigentumsvermerke erhalten bleiben, vervielfältigen, weitergeben und anzeigen. Sie dürfen diese Veröffentlichungen oder Teile dieser Veröffentlichungen ohne ausdrückliche Genehmigung von IBM außerhalb Ihres Unternehmens nicht vervielfältigen, weitergeben, anzeigen oder abgeleitete Werke davon erstellen.

Rechte: Abgesehen von den hier gewährten Berechtigungen erhalten Sie keine weiteren Berechtigungen, Lizenzen oder Rechte (veröffentlicht oder stillschweigend) in Bezug auf die Veröffentlichungen oder darin enthaltene Informationen, Daten, Software oder geistiges Eigentum.

IBM behält sich das Recht vor, die in diesem Dokument gewährten Berechtigungen nach eigenem Ermessen zurückzuziehen, wenn sich die Nutzung der Veröffentlichungen für IBM als nachteilig erweist oder wenn die obigen Nutzungsbestimmungen nicht genau befolgt werden.

Sie dürfen diese Informationen nur in Übereinstimmung mit allen anwendbaren Gesetzen und Vorschriften, einschließlich aller US-amerikanischen Exportgesetze und Verordnungen, herunterladen und exportieren.

IBM übernimmt keine Gewährleistung für den Inhalt dieser Informationen. Diese Veröffentlichungen werden auf der Grundlage des gegenwärtigen Zustands (auf "as-is"-Basis) und ohne eine ausdrückliche oder stillschweigende Gewährleistung für die Handelsüblichkeit, die Verwendungsfähigkeit oder die Freiheit der Rechte Dritter zur Verfügung gestellt.

IBM Marken: IBM, das IBM Logo und ibm.com sind Marken oder eingetragene Marken der International Business Machines Corporation. Weitere Produkt- oder Servicenamen können Marken von oder anderen Herstellern sein. IBM oder anderen Herstellern sein. Eine aktuelle Liste der IBM Marken finden Sie auf der Webseite www.ibm.com/legal/copytrade.shtml.

Anhang B. Bemerkungen

Die vorliegenden Informationen wurden für Produkte und Services entwickelt, die auf dem deutschen Markt angeboten werden. Die Informationen über Produkte anderer Hersteller als IBM basieren auf den zum Zeitpunkt der ersten Veröffentlichung dieses Dokuments verfügbaren Informationen und können geändert werden.

Möglicherweise bietet IBM die in dieser Dokumentation beschriebenen Produkte, Services oder Funktionen in anderen Ländern nicht an. Informationen über die gegenwärtig im jeweiligen Land verfügbaren Produkte und Services sind beim zuständigen IBM Ansprechpartner erhältlich. Hinweise auf IBM Lizenzprogramme oder andere IBM Produkte bedeuten nicht, dass nur Programme, Produkte oder Services von IBM verwendet werden können. Anstelle der IBM Produkte, Programme oder Services können auch andere, ihnen äquivalente Produkte, Programme oder Services verwendet werden, solange diese keine gewerblichen oder anderen Schutzrechte von IBM verletzen. Die Verantwortung für den Betrieb von Produkten, Programmen und Services anderer Anbieter liegt beim Kunden.

Für in diesem Handbuch beschriebene Erzeugnisse und Verfahren kann es IBM Patente oder Patentanmeldungen geben. Mit der Auslieferung dieses Handbuchs ist keine Lizenzierung dieser Patente verbunden. Lizenzanforderungen sind schriftlich an folgende Adresse zu richten (Anfragen an diese Adresse müssen auf Englisch formuliert werden):

IBM Director of Licensing
IBM Europe, Middle East & Africa
Tour Descartes
2, avenue Gambetta
92066 Paris La Defense
France

Trotz sorgfältiger Bearbeitung können technische Ungenauigkeiten oder Druckfehler in dieser Veröffentlichung nicht ausgeschlossen werden. Die hier enthaltenen Informationen werden in regelmäßigen Zeitabständen aktualisiert und als Neuausgabe veröffentlicht. IBM kann ohne weitere Mitteilung jederzeit Verbesserungen und/oder Änderungen an den in dieser Veröffentlichung beschriebenen Produkten und/oder Programmen vornehmen.

Verweise in diesen Informationen auf Websites anderer Anbieter werden lediglich als Service für den Kunden bereitgestellt und stellen keinerlei Billigung des Inhalts dieser Websites dar. Das über diese Websites verfügbare Material ist nicht Bestandteil des Materials für dieses IBM Produkt. Die Verwendung dieser Websites geschieht auf eigene Verantwortung.

Werden an IBM Informationen eingesandt, können diese beliebig verwendet werden, ohne dass eine Verpflichtung gegenüber dem Einsender entsteht.

Lizenznehmer des Programms, die Informationen zu diesem Produkt wünschen mit der Zielsetzung: (i) den Austausch von Informationen zwischen unabhängig voneinander erstellten Programmen und anderen Programmen (einschließlich des vorliegenden Programms) sowie (ii) die gemeinsame Nutzung der ausgetauschten Informationen zu ermöglichen, wenden sich an folgende Adresse:

IBM Canada Limited
U59/3600
3600 Steeles Avenue East
Markham, Ontario L3R 9Z7
CANADA

Die Bereitstellung dieser Informationen kann unter Umständen von bestimmten Bedingungen - in einigen Fällen auch von der Zahlung einer Gebühr - abhängig sein.

Die Lieferung des im Dokument aufgeführten Lizenzprogramms sowie des zugehörigen Lizenzmaterials erfolgt auf der Basis der IBM Rahmenvereinbarung bzw. der Allgemeinen Geschäftsbedingungen von IBM, der IBM Internationalen Nutzungsbedingungen für Programmpakete oder einer äquivalenten Vereinbarung.

Alle in diesem Dokument enthaltenen Leistungsdaten stammen aus einer kontrollierten Umgebung. Die Ergebnisse, die in anderen Betriebsumgebungen erzielt werden, können daher erheblich von den hier erzielten Ergebnissen abweichen. Einige Daten stammen möglicherweise von Systemen, deren Entwicklung noch nicht abgeschlossen ist. Eine Gewährleistung, dass diese Daten auch in allgemein verfügbaren Systemen erzielt werden, kann nicht gegeben werden. Darüber hinaus wurden einige Daten unter Umständen durch Extrapolation berechnet. Die tatsächlichen Ergebnisse können davon abweichen. Benutzer dieses Dokuments sollten die entsprechenden Daten in ihrer spezifischen Umgebung prüfen.

Alle Informationen zu Produkten anderer Anbieter stammen von den Anbietern der aufgeführten Produkte, deren veröffentlichen Ankündigungen oder anderen allgemein verfügbaren Quellen. IBM hat diese Produkte nicht getestet und kann daher keine Aussagen zu Leistung, Kompatibilität oder anderen Merkmalen machen. Fragen zu den Leistungsmerkmalen von Produkten anderer Anbieter sind an den jeweiligen Anbieter zu richten.

Aussagen über Pläne und Absichten von IBM unterliegen Änderungen oder können zurückgenommen werden und repräsentieren nur die Ziele von IBM.

Diese Veröffentlichung kann Beispiele für Daten und Berichte des alltäglichen Geschäftsablaufes enthalten. Sie sollen nur die Funktionen des Lizenzprogramms illustrieren; sie können Namen von Personen, Firmen, Marken oder Produkten enthalten. Alle diese Namen sind frei erfunden; Ähnlichkeiten mit tatsächlichen Namen und Adressen sind rein zufällig.

COPYRIGHTLIZENZ:

Diese Veröffentlichung enthält Musteranwendungsprogramme, die in Quellensprache geschrieben sind und Programmiertechniken in verschiedenen Betriebsumgebungen veranschaulichen. Sie dürfen diese Musterprogramme kostenlos kopieren, ändern und verteilen, wenn dies zu dem Zweck geschieht, Anwendungsprogramme zu entwickeln, zu verwenden, zu vermarkten oder zu verteilen, die mit der Anwendungsprogrammierschnittstelle für die Betriebsumgebung konform sind, für die diese Musterprogramme geschrieben werden. Diese Beispiele wurden nicht unter allen denkbaren Bedingungen getestet. Daher kann IBM die Zuverlässigkeit, Wartungsfreundlichkeit oder Funktion dieser Programme weder zusagen noch gewährleisten. Die Musterprogramme werden ohne Wartung (auf "as-is"-Basis) und ohne jegliche Gewährleistung zur Verfügung gestellt. IBM haftet nicht für Schäden, die durch Verwendung der Musterprogramme entstehen.

Kopien oder Teile der Musterprogramme bzw. daraus abgeleiteter Code müssen folgenden Copyrightvermerk beinhalten:

© (*Name Ihrer Firma*) (*Jahr*). Teile des vorliegenden Codes wurden aus Musterprogrammen der IBM Corp. abgeleitet. © Copyright IBM Corp. *_Jahr/Jahre angeben_*. Alle Rechte vorbehalten.

Marken

IBM, das IBM Logo und ibm.com sind Marken oder eingetragene Marken der IBM Corporation in den USA und/oder anderen Ländern. Weitere Produkt- oder Servicenamen können Marken von oder anderen Herstellern sein. IBM oder anderen Herstellern sein. Eine aktuelle Liste der IBM Marken finden Sie auf der Webseite „Copyright and trademark information" unter www.ibm.com/legal/copytrade.shtml.

Die folgenden Namen sind Marken oder eingetragene Marken anderer Unternehmen.

- Linux ist eine eingetragene Marke von Linus Torvalds in den USA und/oder anderen Ländern.
- Java und alle auf Java basierenden Marken und Logos sind Marken oder eingetragene Marken von Oracle und/oder ihren verbundenen Unternehmen.
- UNIX ist eine eingetragene Marke von The Open Group in den USA und anderen Ländern.
- Intel, das Intel-Logo, Intel Inside, Intel Inside logo, Celeron, Intel SpeedStep, Itanium und Pentium sind Marken oder eingetragene Marken der Intel Corporation oder deren Tochtergesellschaften in den USA und anderen Ländern.
- Microsoft, Windows, Windows NT und das Windows-Logo sind Marken der Microsoft Corporation in den USA und/oder anderen Ländern.

Weitere Unternehmens-, Produkt- oder Servicenamen können Marken anderer Hersteller sein.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

SC12-4673-00

Spine information:

IBM DB2 10.1 for Linux, UNIX and Windows

Datenbankverwaltung - Konzepte und Konfiguration - Referenzinformationen