IBM DB2 Universal Database

Systemverwaltung: Optimierung

Version 8.2

SC12-3058-01

IBM DB2 Universal Database

Systemverwaltung: Optimierung

Version 8.2

SC12-3058-01

Anmerkung

Vor Verwendung dieser Informationen und des darin beschriebenen Produkts sollten die allgemeinen Informationen unter *Bemerkungen* gelesen werden.

- Die IBM Homepage finden Sie im Internet unter: **ibm.com**
- IBM und das IBM Logo sind eingetragene Marken der International Business Machines Corporation.
- Das e-business-Symbol ist eine Marke der International Business Machines Corporation.
- Infoprint ist eine eingetragene Marke der IBM.
- ActionMedia, LANDesk, MMX, Pentium und ProShare sind Marken der Intel Corporation in den USA und/oder anderen Ländern.
- C-bus ist eine Marke der Corollary, Inc. in den USA und/oder anderen Ländern.
- Java und alle auf Java basierenden Marken und Logos sind Marken der Sun Microsystems, Inc. in den USA und/oder anderen Ländern.
- Microsoft Windows, Windows NT und das Windows-Logo sind Marken der Microsoft Corporation in den USA und/oder anderen Ländern.
- PC Direct ist eine Marke der Ziff Communications Company in den USA und/oder anderen Ländern.
- SET und das SET-Logo sind Marken der SET Secure Electronic Transaction LLC.
- UNIX ist eine eingetragene Marke der Open Group in den USA und/oder anderen Ländern.
- Marken anderer Unternehmen/Hersteller werden anerkannt.

Diese Veröffentlichung ist eine Übersetzung des Handbuchs
IBM DB2 Universal Database Administration Guide: Performance,
IBM Form GC09-4821-01

herausgegeben von International Business Machines Corporation, USA

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:
SW TSC Germany
Kst. 2877
April 2004

Inhaltsverzeichnis

Zu diesem Handbuch

Das Handbuch zur Systemverwaltung bietet in seinen drei Bänden Informationen, die zur Verwendung und Verwaltung der DB2-Produkte für Verwaltungssysteme für relationale Datenbanken (RDBMS) benötigt werden:

- Informationen zum Datenbankentwurf (im Band *Systemverwaltung: Konzept*)
- Informationen zur Implementierung und Verwaltung von Datenbanken (im Band *Systemverwaltung: Implementierung*)
- Informationen zur Konfiguration und Optimierung der Datenbankumgebung zum Zweck der Leistungsverbesserung (im Band *Systemverwaltung: Optimierung*)

Viele der in diesem Handbuch beschriebenen Operationen können mit Hilfe verschiedener Schnittstellen durchgeführt werden:

- Der **Befehlszeilenprozessor** ermöglicht Ihnen den Zugriff auf Datenbanken und deren Bearbeitung über eine grafische Schnittstelle. Von dieser Schnittstelle aus können Sie SQL-Anweisungen und DB2-Dienstprogrammfunktionen ausführen. Die Mehrzahl der Beispiele in diesem Handbuch zeigt die Verwendung dieser Schnittstelle. Weitere Informationen zur Verwendung des Befehlszeilenprozessors finden Sie im Handbuch *Command Reference*.
- Die **Anwendungsprogrammierschnittstelle** ermöglicht Ihnen die Ausführung von DB2-Dienstprogrammfunktionen innerhalb eines Anwendungsprogramms. Weitere Informationen zur Verwendung der Anwendungsprogrammierschnittstelle finden Sie im Handbuch *Administrative API Reference*.
- Die **Steuerzentrale** ermöglicht Ihnen die Ausführung von Verwaltungsaufgaben, z. B. das Konfigurieren des Systems, die Verwaltung von Verzeichnissen, das Sichern und Wiederherstellen des Systems, die zeitliche Terminierung von Jobs und die Verwaltung von Medien, über eine grafische Schnittstelle. Die Steuerzentrale enthält außerdem eine Replikationsverwaltung, mit der die Replikation von Daten zwischen den Systemen eingerichtet werden kann. Darüber hinaus ermöglicht die Steuerzentrale das Ausführen von DB2-Dienstprogrammfunktionen über eine grafische Benutzerschnittstelle. Je nach Plattform gibt es unterschiedliche Möglichkeiten, die Steuerzentrale aufzurufen. Geben Sie zum Beispiel den Befehl `db2cc` in der Befehlszeile ein, wählen Sie das Symbol der Steuerzentrale im DB2-Ordner aus oder verwenden Sie auf Windows-Plattformen das Menü **Start**. Wenn Sie eine einführende Hilfe benötigen, wählen Sie **Erste Schritte** im Menü **Hilfe** des Fensters der Steuerzentrale aus. Das Tool **Visual Explain** wird über die Steuerzentrale aufgerufen.

  Die Steuerzentrale ist in drei Sichten verfügbar:

  – Basissicht. Diese Sicht zeigt die zentralen DB2 UDB-Funktionen für wesentliche Objekte wie Datenbanken, Tabellen und gespeicherte Prozeduren.
  – Erweiterte Sicht. Diese Sicht enthält alle verfügbaren Objekte und Aktionen. Verwenden Sie diese Sicht, wenn Sie in einer Unternehmensumgebung arbeiten und eine Verbindung zu DB2 für z/OS oder IMS herstellen wollen.
  – Angepasste Sicht. Diese Sicht gibt Ihnen die Möglichkeit, die Objektbaumstruktur und die Objektaktionen individuell anzupassen.

Es gibt darüber hinaus noch weitere Tools, die Sie zur Durchführung von Verwaltungsaufgaben verwenden können. Zu diesen Schnittstellen gehören:

- Der Befehlseditor, der die Befehlszentrale ersetzt und zum Generieren, Editieren, Ausführen und Manipulieren von SQL-Anweisungen, IMS- und DB2-Befehle,

sowie zum Arbeiten mit der resultierenden Ausgabe und zum Anzeigen einer grafischen Darstellung des Zugriffsplans für mit EXPLAIN bearbeitete SQL-Anweisungen dient.
- Die Entwicklungszentrale, die eine Unterstützung für native gespeicherte SQL-Prozeduren des Persistent Storage Module (PSM), für gespeicherte Java-Prozeduren für iSeries Version 5 Release 3 und spätere Versionen sowie für benutzerdefinierte Funktionen (UDFs) und strukturierte Typen bereitstellt.
- Die Diagnosezentrale, die ein Tool zur Verfügung stellt, das Datenbankadministratoren (DBA) bei der Lösung von Problemen mit der Leistung und der Ressourcenzuordnung unterstützt.
- Das Programm „Tools - Einstellungen", das zur Änderung von Einstellungen für die Steuerzentrale, die Diagnosezentrale und die Replikationszentrale dient.
- Das Journal, das zur zeitlichen Terminierung von Jobs dient, die im nicht überwachten Modus ausgeführt werden sollen.
- Die Data Warehouse-Zentrale, die zur Verwaltung von Warehouse-Objekten dient.

Zielgruppe

Dieses Handbuch ist in erster Linie für Datenbankadministratoren, Systemadministratoren, Sicherheitsadministratoren und Systembediener gedacht, die eine Datenbank für den Zugriff durch lokale oder ferne Clients entwerfen, implementieren und pflegen müssen. Es wendet sich auch an Programmierer und andere Benutzer, die Kenntnisse über die Verwaltung und Bedienung des relationalen Datenbankverwaltungssystems von DB2 Universal Database™ (DB2 UDB) benötigen.

Aufbau dieses Handbuchs

Das vorliegende Handbuch enthält Informationen zu folgenden Hauptthemen:

Einführung in die Optimierung

- Kapitel 1, „Einführung in die Optimierung", enthält eine Einführung in die Konzepte und Überlegungen zur Verwaltung und Optimierung der Leistung von DB2 UDB.
- Kapitel 2, „Architektur und Prozesse", stellt die zugrundeliegende Architektur und die Prozesse von DB2 Universal Database vor.

Optimieren der Anwendungsleistung

- Kapitel 3, „Überlegungen zu Anwendungen", beschreibt einige Techniken zur Verbesserung der Datenbankleistung beim Entwurf der Anwendungen.
- Kapitel 4, „Überlegungen zur Umgebung", beschreibt einige Techniken zur Verbesserung der Datenbankleistung bei der Einrichtung der Datenbankumgebung.
- Kapitel 5, „Systemkatalogstatistiken", beschreibt einige Techniken zur Erfassung von Statistiken über die Daten und ihre Verwendung zur Gewährleistung einer optimalen Leistung.
- Kapitel 6, „Der SQL-Compiler", beschreibt die Verarbeitung einer SQL-Anweisung durch den SQL-Compiler bei der Kompilierung.
- Kapitel 7, „Die SQL-EXPLAIN-Einrichtung", beschreibt die Einrichtung EXPLAIN, die Ihnen ermöglicht, die Pfade und Methoden anzuzeigen, die der SQL-Compiler ausgewählt hat, um auf die Daten zuzugreifen.

Optimieren und Konfigurieren des Systems

- Kapitel 8, „Leistung bei der Ausführung", gibt einen Überblick über die Verwendung von Speicher durch den Datenbankmanager und enthält Informationen zu weiteren Faktoren, die sich auf die Leistung zur Laufzeit auswirken.
- Kapitel 9, „Verwenden von Governor", bietet eine Einführung in die Verwendung des Programms Governor, mit dem einige Aspekte der Datenbankverwaltung gesteuert werden können.
- Kapitel 10, „Skalieren der Konfiguration", enthält einige Informationen und Hinweise, die bei einer Vergrößerung von Datenbanksystemen von Bedeutung sind.
- Kapitel 11, „Umverteilen von Daten auf Datenbankpartitionen", behandelt die Punkte, die in einer Umgebung mit partitionierten Datenbanken bei der Neuverteilung von Daten auf die Partitionen zu beachten sind.
- Kapitel 12, „Durchführen von Vergleichstests", gibt einen Überblick über Vergleichstests und behandelt verschiedene Aspekte ihrer Durchführung.
- Kapitel 13, „Konfigurieren von DB2", behandelt die Konfigurationsdateien für den Datenbankmanager und die Datenbanken sowie die Werte für die Konfigurationsparameter für den Datenbankmanager, die Datenbanken und den DB2-Verwaltungsserver (DAS).

Anhänge

- Anhang A, „DB2-Registriervariablen und DB2-Umgebungsvariablen", enthält Werte für die Profilregistrierdatenbank und für Umgebungsvariablen.
- Anhang B, „EXPLAIN-Tabellen", enthält Informationen zu den Tabellen, die von der DB2-EXPLAIN-Einrichtung verwendet werden, und beschreibt die Erstellung dieser Tabellen.
- Anhang C, „SQL-EXPLAIN-Tools", enthält Informationen zur Verwendung der DB2-EXPLAIN-Programme: db2expln und dynexpln.
- Anhang D, „db2exfmt - Formatierungstool für EXPLAIN-Tabellen", enthält Informationen dazu, wie der Inhalt von EXPLAIN-Tabellen formatiert wird.

Kurzübersicht über die anderen Bände des Handbuchs zur Systemverwaltung

Systemverwaltung: Konzept

Der Band *Systemverwaltung: Konzept* behandelt den Datenbankentwurf. Er enthält Themen zum logischen und physischen Entwurf sowie zu verteilten Transaktionen. Die einzelnen Kapitel und Anhänge des Bandes werden im Folgenden kurz vorgestellt:

Datenbankkonzepte

- Das Kapitel zu allgemeinen Konzepten relationaler Datenbanken enthält eine Übersicht über Datenbankobjekte, einschließlich Wiederherstellungsobjekten, Speicherobjekten und Systemobjekten.
- Das Kapitel über parallele Datenbanksysteme enthält eine Einführung in die Arten von Parallelität, die mit Hilfe von DB2 implementiert werden können.
- Das Kapitel über Data Warehouses enthält eine Übersicht über den Einsatz von Data Warehouses und Data Warehouse-Funktionen.

Datenbankentwurf

- Das Kapitel zum Entwurf des logischen Datenbankaufbaus behandelt die Konzepte und Richtlinien für den logischen Entwurf einer Datenbank.
- Das Kapitel zum Entwurf der physischen Datenbank behandelt die Richtlinien für den physischen Entwurf einer Datenbank und enthält Überlegungen im Hinblick auf die Datenspeicherung.
- Das Kapitel zum Entwerfen verteilter Datenbanken beschreibt die Möglichkeiten des Zugriffs auf mehrere Datenbanken in einer einzigen Transaktion.
- Das Kapitel zum Entwerfen für Transaktionsmanager behandelt die Verwendung von Datenbanken in einer Umgebung für verteilte Transaktionsverarbeitung.

Anhänge

- Der Anhang über Inkompatibilitäten zwischen Releases stellt die Inkompatibilitäten dar, die von Version 7 und Version 8 eingeführt werden, und weist auf zukünftige Inkompatibilitäten hin, auf die geachtet werden sollte.
- Der Anhang zur Unterstützung von Landessprachen beschreibt die DB2-Unterstützung von Landessprachen und enthält Informationen zu Gebieten, Sprachen und Codepages.
- Der Anhang zum Aktivieren der Unterstützung großer Seiten in einer 64-Bit-Umgebung (AIX) erläutert die Unterstützung für die Seitengröße von 16 MB sowie die Aktivierung dieser Unterstützung.

Systemverwaltung: Implementierung

Der Band *Systemverwaltung: Implementierung* behandelt die Implementierung des entwickelten Datenbankentwurfs. Die einzelnen Kapitel und Anhänge des Bandes werden im Folgenden kurz vorgestellt:

Implementieren des Datenbankentwurfs

- Das Kapitel ″Vor dem Erstellen einer Datenbank″ beschreibt die Voraussetzungen für die Erstellung einer Datenbank und der Objekte innerhalb einer Datenbank.
- Das Kapitel über das Erstellen und Verwenden eines DB2-Verwaltungsservers (DAS) erläutert, was ein DAS ist, wie er erstellt wird und wie er verwendet wird.
- Das Kapitel über das Erstellen einer Datenbank beschreibt die Aufgaben der Erstellung einer Datenbank sowie der zugehörigen Datenbankobjekte.
- Das Kapitel über das Erstellen von Tabellen und anderen zugehörigen Tabellenobjekten beschreibt die Erstellung von Tabellen mit spezifischen Merkmalen bei der Implementierung eines Datenbankentwurfs.
- Das Kapitel über das Ändern einer Datenbank behandelt, was vor der Änderung einer Datenbank zu tun ist und welche Aufgaben im Zusammenhang mit dem Ändern oder Löschen einer Datenbank oder zugehöriger Datenbankobjekte erledigt werden müssen.
- Das Kapitel über das Ändern von Tabellen und anderen zugehörigen Tabellenobjekten beschreibt das Löschen von Tabellen bzw. das Ändern der diesen Tabellen zugeordneten spezifischen Merkmale. Das Löschen und Ändern zusammengehöriger Tabellenobjekte wird ebenfalls an dieser Stelle erläutert.

Datenbanksicherheit

- Das Kapitel zur Steuerung des Datenbankzugriffs beschreibt die Möglichkeiten zur Steuerung des Zugriffs auf die Ressourcen einer Datenbank.
- Das Kapitel zur Protokollierung von DB2-Aktivitäten beschreibt Methoden zur Erkennung und Überwachung unerwünschten bzw. unvorhergesehenen Zugriffs auf Daten.

Anhänge

- Der Anhang zu den Namenskonventionen enthält die Regeln, die bei der Benennung von Datenbanken und Objekten zu beachten sind.
- Der Anhang zur Verwendung der automatischen Clientweiterleitung erläutert die automatische Weiterleitung von Clientanwendungen sowie die Aktivierung dieser Unterstützung.
- Der Anhang zur Verwendung der LDAP-Verzeichnisservices (Lightweight Directory Access Protocol) enthält Informationen zu den Einsatzmöglichkeiten der LDAP-Verzeichnisservices.
- Der Anhang über das Absetzen von Befehlen an mehrere Datenbankpartitionen behandelt die Verwendung der Shellprozeduren *db2_all* und *rah* zum Senden von Befehlen an alle Partitionen in einer Umgebung mit partitionierten Datenbanken.
- Der Anhang zur Unterstützung für Windows Management Instrumentation (WMI) beschreibt, in welcher Form DB2 diesen Verwaltungsinfrastrukturstandard zur Integration verschiedener als Hard- oder Software implementierter Verwaltungssysteme unterstützt. Darüber hinaus wird die Integration von DB2 in WMI erläutert.
- Der Anhang zur Verwendung der Windows NT-Sicherheit beschreibt, wie DB2 Universal Database mit der Windows NT-Sicherheit arbeitet.
- Der Anhang zur Verwendung des Windows-Systemmonitors enthält Informationen über das Registrieren von DB2 im Windows NT-Systemmonitor und über die Nutzung der Leistungsinformationen.
- Der Anhang zur Verwendung von Windows-Datenbankpartitionsservern enthält Informationen über die verfügbaren Dienstprogramme zur Arbeit mit Datenbankpartitionsservern unter Windows NT oder Windows 2000.
- Der Anhang zur Konfiguration mehrerer logischer Knoten beschreibt, wie mehrere logische Knoten in einer Umgebung mit partitionierten Datenbanken konfiguriert werden.
- Der Anhang zur Erweiterung der Steuerzentrale enthält Informationen zur Erweiterung der Steuerzentrale durch Hinzufügen neuer Knöpfe in der Menüleiste, einschließlich neuer Aktionen, Hinzufügen neuer Objektdefinitionen und Hinzufügen neuer Aktionsdefinitionen.

Anmerkung: Aus diesem Handbuch wurden zwei Kapitel herausgenommen.

Alle Informationen zu den DB2-Dienstprogrammen für das Versetzen von Daten sowie die vergleichbaren Themen aus den Handbüchern *Command Reference* und *Administrative API Reference* wurden in *Dienstprogramme für das Versetzen von Daten Handbuch und Referenz* zusammengefasst.

Informationen zu diesen Themen finden Sie ausschließlich in *Dienstprogramme für das Versetzen von Daten Handbuch und Referenz*.

Weitere Informationen zur Replikation von Daten finden Sie in *IBM DB2 Information Integrator SQL Replication Handbuch und Referenz*.

Alle Informationen über die Methoden und Tools zur Sicherung und Wiederherstellung von Daten sowie die vergleichbaren Themen aus den Handbüchern *Command Reference* und *Administrative API Reference* wurden in *Datenwiederherstellung und hohe Verfügbarkeit Handbuch und Referenz* zusammengefasst.

Informationen zu diesen Themen finden Sie ausschließlich in *Datenwiederherstellung und hohe Verfügbarkeit Handbuch und Referenz*.

Teil 1. Einführung in die Optimierung

Kapitel 1. Einführung in die Optimierung

Die Abschnitte in diesem Kapitel beschreiben die Leistungsoptimierung und enthalten einige Empfehlungen zu folgenden Zwecken:

- Erstellen eines Plans zur Leistungsüberwachung und -optimierung
- Verwenden von Benutzerinformationen zu Leistungsproblemen
- Schnelles Einsteigen in die ersten Schritte der Leistungsoptimierung

Elemente der Leistung

Als *Leistung* wird die Art und Weise bezeichnet, wie ein Computersystem unter einer bestimmten Auslastung arbeitet. Die Leistung wird an der Antwortzeit, am Durchsatz und an der Verfügbarkeit gemessen. Außerdem wird die Leistung von folgenden Faktoren beeinflusst:

- Von den im System verfügbaren Ressourcen
- Von der Auslastung und vom Ausmaß der gemeinsamen Nutzung dieser Ressourcen

Im Allgemeinen optimieren Sie Ihr System, um das Kosten-Nutzen-Verhältnis zu verbessern. Die folgenden speziellen Optimierungsziele können dabei verfolgt werden:

- Verarbeiten einer größeren oder anspruchsvolleren Arbeitsbelastung ohne steigende Verarbeitungskosten

  Zum Beispiel zur Steigerung der Auslastung ohne neue Hardware erwerben oder mehr Prozessorzeit in Kauf nehmen zu müssen.
- Erreichen schnellerer Systemantwortzeiten bzw. eines höheren Durchsatzes ohne Steigerung der Verarbeitungskosten
- Reduzieren von Verarbeitungskosten ohne negative Auswirkungen für Ihre Benutzer

Die Übersetzung von Leistung vom technischen Aspekt in den wirtschaftlichen Aspekt ist schwierig. Eine Leistungsoptimierung kostet natürlich Geld in Form von Zeitaufwand für Personal und Prozessorzeit, so dass vor einem Optimierungsprojekt die Kosten gegen die möglichen Vorteilsgewinne abgewogen werden müssen. Einige dieser Vorteile sind konkret messbar:

- Effizientere Ressourcennutzung
- Die Möglichkeit, weitere Benutzer dem System hinzuzufügen

Andere Vorteile, wie größere Zufriedenheit seitens der Benutzer aufgrund schnellerer Antwortzeiten, sind weniger fassbar. Jedoch sollten alle diese Vorteile in die Überlegungen mit einbezogen werden.

Zugehörige Konzepte:

- „Richtlinien zur Leistungsoptimierung" auf Seite 4
- „Einstiegstipps für die Leistungsoptimierung" auf Seite 7

Zugehörige Tasks:

- „Entwickeln eines Leistungsverbesserungsprozesses" auf Seite 5

Richtlinien zur Leistungsoptimierung

Die folgenden Richtlinien sind als Hilfe für die Entwicklung eines allgemeinen Ansatzes zur Leistungsoptimierung zu verstehen.

Behalten Sie das Gesetz der abnehmenden Ertragsgewinne im Hinterkopf: Die größten Leistungsvorteile werden in der Regel durch die ersten Maßnahmen erzielt. Weitere Änderungen erbringen im Allgemeinen immer kleinere Vorteile und erfordern immer höheren Aufwand.

Optimieren Sie nicht nur des Optimierens wegen: Optimieren Sie, um erkannten Engpässen abzuhelfen. Das Optimieren von Ressourcen, die nicht den Hauptgrund für Leistungsprobleme darstellen, hat wenig oder gar keine Wirkung auf Antwortzeiten, solange Sie nicht die wichtigeren Probleme behoben haben, und kann tatsächlich eine nachfolgende Optimierungsarbeit erschweren. Wenn es ein bedeutendes Verbesserungspotenzial gibt, liegt es in der Verbesserung der Leistung von Ressourcen, die wichtige Faktoren in der Antwortzeit bilden.

Betrachten Sie das gesamte System: Ein Parameter bzw. System lässt sich nicht isoliert optimieren. Bevor Sie Anpassungen vornehmen, überlegen Sie, wie sich diese Anpassungen auf das System als Ganzes auswirken werden.

Ändern Sie jeweils nur einen Parameter gleichzeitig: Ändern Sie nicht mehrere Parameter zur Leistungsoptimierung in einem Schritt. Selbst wenn Sie sich sicher sind, dass alle Änderungen vorteilhaft sind, haben Sie hinterher keine Möglichkeit, den Beitrag jeder Änderung zu bewerten. Darüber hinaus können Sie bei gleichzeitiger Änderung mehrerer Parameter den erzielten Vorteil nicht effektiv den möglicherweise in Kauf genommenen Einbußen gegenüberstellen. Von jeder Anpassung eines Parameters zur Optimierung eines Bereichs ist fast immer auch mindestens ein anderer Bereich betroffen, der vorher vielleicht nicht bedacht wurde. Wenn Sie jeweils nur einen Parameter ändern, haben Sie einen Vergleichspunkt und können feststellen, ob die Änderung die gewünschte Wirkung hat.

Führen Sie Messungen und Neukonfigurierungen nach Ebenen durch: Aus denselben Gründen, aus denen nur jeweils ein Parameter geändert werden soll, empfiehlt sich auch, die einzelnen Ebenen des Systems getrennt zu optimieren. Die folgende Liste von Ebenen innerhalb eines Systems kann Ihnen dabei als Richtlinie dienen:

- Hardware
- Betriebssystem
- Anwendungsserver und -requester
- Datenbankmanager
- SQL-Anweisungen
- Anwendungsprogramme

Prüfen Sie auf Hardware- und Softwareprobleme: Einige Leistungsprobleme können vielleicht durch Wartung der Hardware oder Korrektur der Software oder durch beides behoben werden. Verwenden Sie nicht zu viel Zeit auf die Überwachung und Optimierung des Systems, wenn eine Hardwarewartung oder Softwarekorrektur dies unnötig machen könnte.

Ermitteln Sie die Ursache eines Problems, bevor Sie Ihre Hardware aufrüsten: Auch wenn es so aussieht, als könnten zusätzliche Speicher- und Prozessorkapazitäten die Leistung sofort verbessern, sollten Sie sich die Zeit nehmen, die

Engpässe zu lokalisieren und zu verstehen. Sie könnten ansonsten Geld für zusätzlichen Plattenspeicher ausgeben und anschließend feststellen, dass Sie nicht über die Prozessorkapazitäten oder die Kanäle verfügen, um den Speicher vorteilhaft zu nutzen.

Bereiten Sie Zurücksetzungsprozeduren vor, bevor Sie mit der Optimierung beginnen: Wie bereits früher erwähnt, können einige Optimierungsmaßnahmen zu unerwarteten Leistungsergebnissen führen. Wenn sich daraus eine schlechtere Leistung ergibt, sollten die Maßnahmen rückgängig gemacht und alternative Optimierungsmaßnahmen versucht werden. Wenn der vorige Stand in einer Weise gesichert wurde, dass er einfach wiederhergestellt werden kann, ist die Rücknahme der nicht korrekten Informationen wesentlich einfacher.

Zugehörige Konzepte:

- „Elemente der Leistung" auf Seite 3
- „Einstiegstipps für die Leistungsoptimierung" auf Seite 7

Zugehörige Tasks:

- „Entwickeln eines Leistungsverbesserungsprozesses" auf Seite 5

Prozess der Leistungsoptimierung

Sie entwickeln einen Plan zur Leistungsüberwachung und -optimierung unter Berücksichtigung der Benutzerangaben und der Grenzen der Optimierung für Ihr System.

Entwickeln eines Leistungsverbesserungsprozesses

Ein Leistungsverbesserungsprozess ist ein iteratives und langfristig angelegtes Verfahren zur Überwachung und Optimierung von Leistungsbereichen. Abhängig von den Überwachungsergebnissen passen Sie und Ihr Optimierungsteam die Konfiguration des Datenbankservers an und nehmen Änderungen an den Anwendungen vor, die den Datenbankserver verwenden.

Gehen Sie bei der Leistungsüberwachung und den Optimierungsentscheidungen von Ihren Kenntnissen über die Arten von Anwendungen, die mit den Daten arbeiten, sowie von den Datenzugriffsmustern aus. Verschiedene Arten von Anwendungen haben unterschiedliche Leistungsanforderungen.

Betrachten Sie die folgende Verfahrensstruktur für den Leistungsverbesserungsprozess als Richtlinie.

Vorgehensweise:

Gehen Sie zur Entwicklung eines Leistungsverbesserungsprozesses wie folgt vor:

1. Definieren Sie Leistungsziele.
2. Legen Sie Leistungsindikatoren für die wichtigsten Leistungsprobleme im System fest.
3. Entwickeln Sie einen Leistungsüberwachungsplan und führen Sie ihn aus.
4. Analysieren Sie die Ergebnisse der Überwachung fortlaufend, um zu ermitteln, welche Ressourcen optimiert werden müssen.
5. Nehmen Sie jeweils nur eine Anpassung vor.

Selbst wenn Sie davon überzeugt sind, dass mehr als eine Ressource eine Optimierung erfordert, oder wenn mehrere Optimierungsoptionen für die zu optimierende Ressource möglich sind, sollten Sie nur eine Änderung pro Optimierungsschritt vornehmen, so dass Sie sicherstellen können, dass Ihre Optimierungsmaßnahmen die gewünschte Wirkung erzielen. An einem bestimmten Punkt lässt sich keine weitere Verbesserung der Leistung durch Optimieren des Datenbankservers und der Anwendungen mehr realisieren. Wenn dieser Punkt erreicht ist, bleibt Ihnen nur die Möglichkeit, Ihre Hardware aufzurüsten.

Zur tatsächlichen Leistungsoptimierung müssen Kompromisslösungen für verschiedene Systemressourcen gefunden werden. Zum Beispiel könnten Sie zur Verbesserung der E/A-Leistung die Pufferpools vergrößern, jedoch benötigen größere Pufferpools mehr Speicher, was wiederum andere Bereiche der Leistung beeinträchtigen könnte.

Zugehörige Konzepte:

- „Elemente der Leistung" auf Seite 3
- „Richtlinien zur Leistungsoptimierung" auf Seite 4
- „Einstiegstipps für die Leistungsoptimierung" auf Seite 7
- „Grenzen der Leistungsoptimierung" auf Seite 7
- „Leistungsinformationen, die Benutzer liefern können" auf Seite 6

Leistungsinformationen, die Benutzer liefern können

Die ersten Anzeichen dafür, dass Ihr System optimiert werden müsste, könnten Klagen von Benutzern sein. Wenn Sie nicht genügend Zeit zur Definition von Leistungszielen sowie zur Überwachung und Optimierung in umfassender Weise haben, können Sie sich mit der Leistung auseinander setzen, indem Sie Ihren Benutzern zuhören. In der Regel können Sie bestimmen, wo mit der Untersuchung eines Problems zu beginnen ist, indem Sie einige einfache Fragen stellen. Sie könnten Ihre Benutzer zum Beispiel Folgendes fragen:

- Was meinen sie mit „langsamer Reaktion"? Heißt dies, um zehn Prozent langsamer, als Sie erwarten, oder um das Zigfache langsamer?
- Wann haben Sie das Problem bemerkt? Tritt es erst seit kurzem auf oder war es immer da?
- Haben andere Benutzer das gleiche Problem? Handelt es sich bei diesen Benutzern um einen oder zwei Einzelpersonen oder um eine ganze Gruppe?
- Wenn eine Gruppe von Benutzern die gleichen Probleme hat, sind sie mit demselben lokalen Netzwerk (LAN) verbunden?
- Scheinen die Probleme mit einem bestimmten Transaktions- oder Anwendungsprogramm zusammenzuhängen?
- Erkennen Sie ein Muster im Auftreten des Problems? Zum Beispiel: Tritt dieses Problem zu einer bestimmten Tageszeit, z. B. in der Mittagspause, auf oder ist es mehr oder weniger permanent spürbar?

Zugehörige Konzepte:

- „Richtlinien zur Leistungsoptimierung" auf Seite 4

Zugehörige Tasks:

- „Entwickeln eines Leistungsverbesserungsprozesses" auf Seite 5

Grenzen der Leistungsoptimierung

Eine Optimierung kann die Effizienz eines Systems nur um einen bestimmten Betrag ändern. Überlegen Sie, wie viel Zeit und Geld Sie in die Optimierung der Systemleistung investieren sollten, und wie viel Aufwand an Zeit und Geld den Benutzern des Systems tatsächlich hilft.

Zum Beispiel kann eine Optimierung häufig die Leistung verbessern, wenn das System auf einen Leistungsengpass stößt. Wenn Ihr System nahe an den Leistungsgrenzen arbeitet und sich die Anzahl von Benutzern am System um zehn Prozent erhöht, verlängern sich die Antwortzeiten wahrscheinlich um wesentlich mehr als zehn Prozent. In dieser Situation müssen Sie feststellen, wie Sie dieser Beeinträchtigung der Leistung durch Optimieren des Systems entgegenwirken können.

Es gibt allerdings einen Punkt, ab dem das Optimieren keine weiteren Leistungsgewinne erzielen kann. An diesem Punkt müssen Sie Ihre Ziele und Erwartungen innerhalb der Grenzen Ihrer Umgebung überprüfen. Wenn Sie bedeutsame Leistungsverbesserungen erreichen wollen, müssen Sie vielleicht mehr Plattenspeicher, eine schnellere CPU, zusätzliche CPUs, mehr Arbeitsspeicher, schnellere Kommunikationsverbindungen oder eine Kombination aus diesen Möglichkeiten hinzufügen.

Zugehörige Konzepte:

- „Verwaltung der Datenbankserverkapazität" auf Seite 331

Zugehörige Tasks:

- „Entwickeln eines Leistungsverbesserungsprozesses" auf Seite 5

Einstiegstipps für die Leistungsoptimierung

Wenn Sie ein neues Exemplar von DB2® starten, ziehen Sie die folgenden Empfehlungen für eine Ausgangskonfiguration in Betracht:

- Verwenden Sie den Konfigurationsadvisor in der Steuerzentrale, um Empfehlungen für sinnvolle Ausgangsstandardwerte für Ihr System zu erhalten. Die Standardwerte, die für DB2 voreingestellt sind, sollten für Ihre spezielle Hardwareumgebung optimiert werden.

  Stellen Sie Informationen zur Hardware an Ihrem Standort zusammen, so dass Sie auf die Fragen des Assistenten antworten können. Sie können die vorgeschlagenen Einstellungen für Konfigurationsparameter sofort anwenden oder den Assistenten eine Prozedur (Script) erstellen lassen, die auf Ihren Antworten beruht, und diese Prozedur später ausführen.

  Diese Prozedur stellt auch eine Liste der am häufigsten optimierten Parameter zur späteren Referenz bereit.
- Verwenden Sie andere Assistenten in der Steuerzentrale und in „Clientkonfiguration - Unterstützung" für leistungsrelevante Verwaltungsaufgaben. Solche Aufgaben sind in der Regel diejenigen, bei denen Sie beträchtliche Leistungsverbesserungen mit geringem Zeitaufwand und wenig Mühe erreichen.

  Andere Assistenten können Ihnen helfen, die Leistung einzelner Tabellen und des allgemeinen Datenzugriffs zu verbessern. Solche Assistenten sind: Datenbank erstellen, Tabelle erstellen, Index erstellen und Aktualisierung auf mehreren Systemen konfigurieren. Die Diagnosezentrale stellt einen Satz von Überwachungs- und Optimierungstools bereit.

- Verwenden Sie das Designadvisor-Tool in der Steuerzentrale oder über den Befehl `db2advis`, um zu ermitteln, welche Indizes, gespeicherten Abfragetabellen, MDC-Tabellen und Datenbankpartitionen die Abfrageleistung verbessern würden.
- Verwenden Sie den Befehl ACTIVATE DATABASE, um Datenbanken zu starten. In einer partitionierten Datenbank aktiviert dieser Befehl die Datenbank in allen Partitionen und vermeidet die Startzeit, die zur Initialisierung der Datenbank erforderlich ist, wenn die erste Anwendung eine Verbindung herstellt.

  Anmerkung: Wenn Sie den Befehl ACTIVATE DATABASE verwenden, müssen Sie die Datenbank mit dem Befehl DEACTIVATE DATABASE herunterfahren. Die letzte Anwendung, die ihre Verbindung zur Datenbank trennt, bewirkt nicht, dass die Datenbank heruntergefahren wird.
- Lesen Sie die Übersichtstabellen, die jeden für den Datenbankmanager und für jede Datenbank verfügbaren Konfigurationsparameter auflisten und kurz beschreiben.

  Diese Übersichtstabellen enthalten eine Spalte, die angibt, ob eine Optimierung des jeweiligen Parameters einen hohen, mittleren, niedrigen oder keinen Einfluss auf die Leistung in positivem oder negativem Sinn hat. Verwenden Sie diese Tabellen, um die Parameter zu ermitteln, die Sie optimieren können, um die größten Leistungsverbesserungen zu erzielen.

Zugehörige Konzepte:

- „Informationen des Datenbanksystemmonitors" auf Seite 310

Zugehörige Referenzen:

- „Konfigurationsparameter - Übersicht" auf Seite 380

Kapitel 2. Architektur und Prozesse

Dieses Kapitel enthält allgemeine Informationen zur Architektur und zum Prozess-Schema von DB2.

Übersicht über die Architektur und Prozesse von DB2

Allgemeine Informationen zur Architektur und zu den Prozessen von DB2® können Ihnen beim Verständnis der detaillierten Informationen zu bestimmten Themen helfen.

Die folgende Abbildung zeigt eine allgemeine Übersicht über die Architektur und die Prozesse für DB2 UDB.

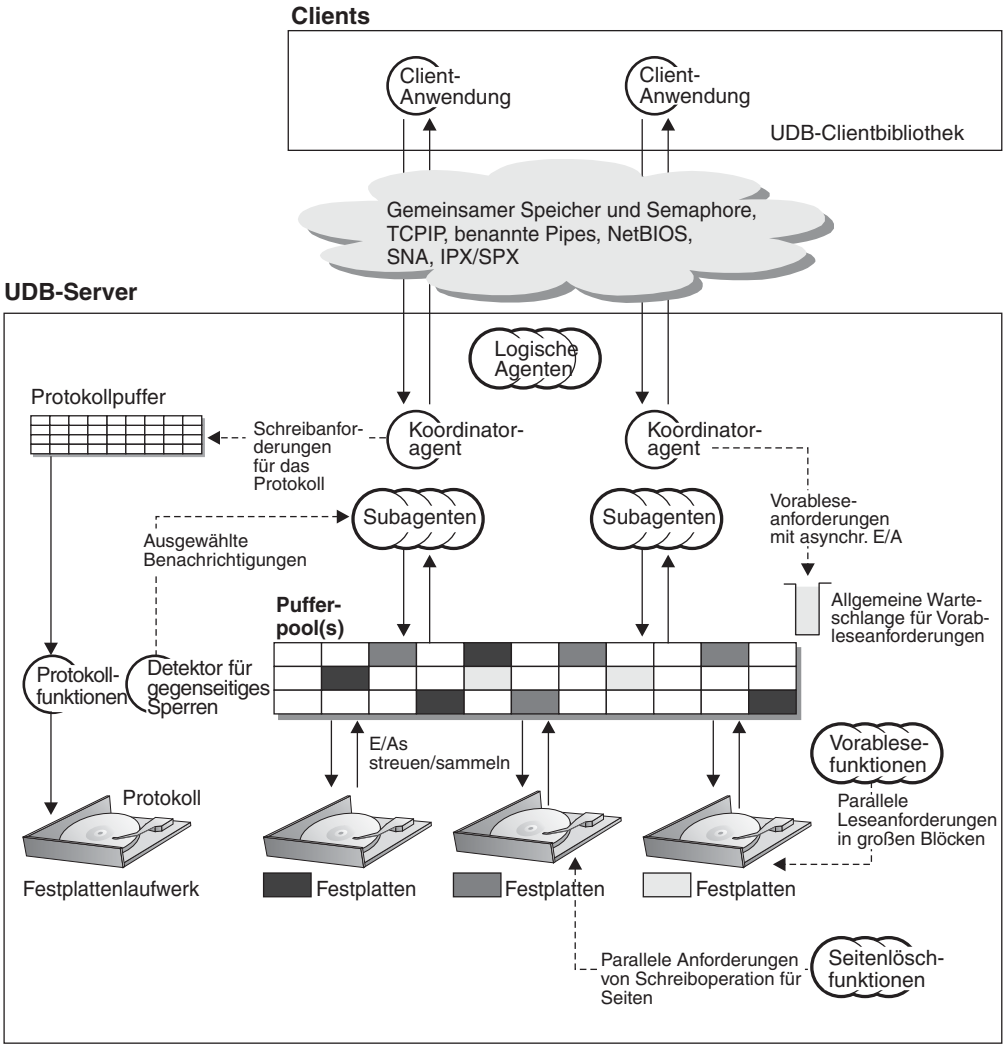


Abbildung 1. Architektur- und Prozessübersicht

Auf der Clientseite werden lokale oder ferne Anwendungen oder beides mit der Clientbibliothek von DB2 Universal Database™ verbunden. Lokale Client kommunizieren über gemeinsamen Speicher und Semaphors, ferne Clients verwenden ein Protokoll wie benannte Pipes (Named Pipes - NPIPE), TCP/IP, NetBIOS oder SNA.

Auf der Serverseite werden die Aktivitäten durch zuteilbare Einheiten der Steuerkomponente, so genannte EDUs (Engine Dispatchable Units), gesteuert. In allen Abbildungen dieses Abschnitts werden EDUs als Kreise oder Gruppen von Kreisen dargestellt. EDUs werden als Threads in einem einzigen Prozess auf Windows®-Plattformen und als Prozesse unter UNIX® implementiert. DB2-Agenten sind die gängigste Art von EDUs. Diese Agenten führen den größten Teil der SQL-Verarbeitung im Auftrag von Anwendungen aus. Vorablesefunktionen und Seitenlöschfunktionen sind weitere häufig genutzte Arten von EDUs.

Die Verarbeitung der Anforderungen einer Clientanwendung kann einer Gruppe von Subagenten übertragen werden. Mehrere Subagenten können zugeordnet werden, wenn die Maschine, auf der sich der Server befindet, mehrere Prozessoren hat oder Teil einer partitionierten Datenbank ist. In einer symmetrischen Multiprozessorumgebung (SMP) können mehrere SMP-Subagenten zum Beispiel die höhere Anzahl der Prozessoren ausnutzen.

Alle Agenten und Subagenten werden mit Hilfe eines Poolfunktionsalgorithmus verwaltet, der die Häufigkeit der Erstellung und Vernichtung von EDUs so gering wie möglich hält.

Pufferpools sind Bereiche des Datenbankserverspeichers, in die Datenbankseiten mit Benutzertabellendaten, Indexdaten und Katalogdaten temporär eingelesen und dort modifiziert werden können. Pufferpools sind eine Schlüsseldeterminante der Datenbankleistung, weil der Zugriff auf Daten im Hauptspeicher wesentlich schneller als auf Daten auf dem Plattenspeicher erfolgen kann. Wenn größere Teile der Daten, die von Anwendungen benötigt werden, in einem Pufferpool vorhanden sind, ist für den Zugriff auf die Daten weniger Zeit erforderlich, als wenn sie auf einer Festplatte gesucht werden müssten.

Die Konfiguration der Pufferpools sowie der EDUs für Vorablesefunktionen und Seitenlöschfunktionen steuert, wie schnell auf Daten zugegriffen werden kann und wie rasch sie Anwendungen verfügbar gemacht werden können.

- **Vorablesefunktionen** rufen Daten von der Festplatte ab und lesen sie in den Pufferpool ein, bevor Anwendungen die Daten benötigen. Beispielsweise müssten Anwendungen, die große Volumina von Daten durchsuchen müssen, darauf warten, dass Daten vom Plattenspeicher in den Pufferpool gelesen werden, wenn es keine Vorablesefunktionen für Daten gäbe. Agenten der Anwendung senden asynchrone Vorableseanforderungen an eine allgemeine Vorablesewarteschlange. Wenn Vorablesefunktionen verfügbar werden, implementieren sie diese Anforderungen. Dabei verwenden sie Eingabeverfahren wie das Lesen großer Blöcke (Big-Block Read) oder gestreutes Lesen (Scatter Read), um die angeforderten Seiten vom Plattenspeicher in den Pufferpool zu laden. Wenn Sie mehrere Plattendatenträger zum Speichern der Datenbankdaten haben, können die Daten über die Plattendatenträger einheitenübergreifend gespeichert werden (Striping). Durch dieses einheitenübergreifende Lesen und Schreiben von Daten können Vorablesefunktionen mehrere Festplatten gleichzeitig zum Abrufen von Daten verwenden.

- **Seitenlöschfunktionen** speichern Daten aus dem Pufferpool zurück auf den Plattenspeicher. Seitenlöschfunktionen sind im Hintergrund aktive EDUs, die unabhängig von den Anwendungsagenten arbeiten. Sie suchen nach Seiten im Pufferpool, die nicht länger benötigt werden, und schreiben die Seiten auf den Plattenspeicher. Seitenlöschfunktionen stellen sicher, dass im Pufferpool Platz für Seiten ist, die von Vorablesefunktionen eingelesen werden.

Ohne diese unabhängigen EDUs für Vorablesezugriffe und Seitenlöschfunktionen müssten Anwendungsagenten die gesamten Lese- und Schreiboperationen für Daten zwischen dem Pufferpool und dem Plattenspeicher selbst durchführen.

Zugehörige Konzepte:

- „Vorablesen von Daten in den Pufferpool" auf Seite 268
- „Gegenseitige Sperren zwischen Anwendungen" auf Seite 11
- „Verzeichnisse und Dateien einer Datenbank" auf Seite 13
- „Protokollverarbeitung" auf Seite 28
- „Aktualisierungsverarbeitung" auf Seite 31
- „Client-/Serververarbeitungsmodell" auf Seite 32
- „Speicherverwaltung" auf Seite 38
- „Verbesserungen des Verbindungskonzentrators für Clientverbindungen" auf Seite 306

Zugehörige Referenzen:

- „max_coordagents - Maximale Anzahl koordinierender Agenten" auf Seite 445
- „max_connections - Maximale Anzahl von Clientverbindungen" auf Seite 444

Gegenseitige Sperren zwischen Anwendungen

Wenn mehrere Anwendungen mit Daten aus der Datenbank arbeiten, kann es zwischen zwei oder mehreren von ihnen zu gegenseitigen Sperren kommen.

Gegenseitige Sperren entstehen, wenn eine Anwendung darauf wartet, dass eine andere Anwendung eine Sperre für Daten freigibt. Die wartenden Anwendungen sperren selbst Daten, die von der jeweils anderen benötigt werden. Dieses gegenseitige Warten darauf, dass die andere Anwendung eine Sperre für Daten freigibt, führt zu gegenseitigem Sperren. Die Anwendungen können unbegrenzt darauf warten, dass eine der Anwendungen die aktive Sperre für die Daten freigibt.

Ein solches gegenseitiges Sperren ist in der folgenden Abbildung dargestellt.

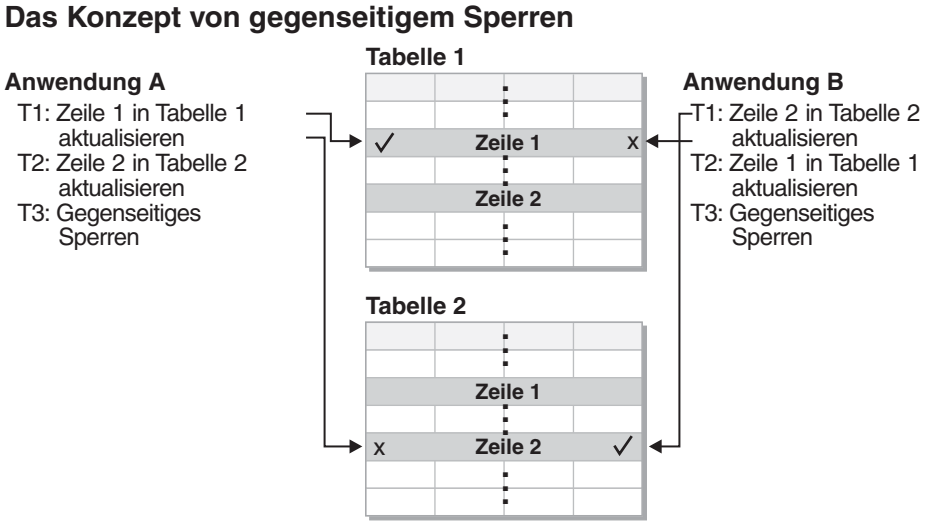


Abbildung 2. Detektor für gegenseitiges Sperren

Da Anwendungen Sperren für Daten, die sie benötigen, nicht von sich aus freigeben, ist ein Detektorprozess erforderlich, der gegenseitige Sperren auflöst und eine Fortsetzung der Anwendungsverarbeitung ermöglicht. Wie der Name schon sagt, überwacht der Detektor für gegenseitiges Sperren die Informationen zu Agenten, die an Sperren warten. Der Detektor für gegenseitiges Sperren wählt eine beliebige der gegenseitig gesperrten Anwendungen aus und gibt die Sperren frei, die momentan von der durch diese Auswahl „betroffenen" Anwendung aktiviert sind. Das Freigeben der Sperren dieser Anwendung führt dazu, dass die Daten wieder verfügbar werden, die von anderen wartenden Anwendungen benötigt werden. Die wartenden Anwendungen können dann auf die erforderlichen Daten zugreifen, um die Transaktionen abzuschließen.

Zugehörige Konzepte:

- „Sperren und Leistung" auf Seite 57
- „Übersicht über die Architektur und Prozesse von DB2" auf Seite 9

Übersicht über den Plattenspeicher

Kenntnisse über die Art und Weise, wie Daten auf Datenträgern gespeichert werden, sind für die E/A-Optimierung hilfreich.

Leistungsfaktoren für Plattenspeichern

Die Hardware, aus der sich Ihr System zusammensetzt, kann die Leistung Ihres Systems beeinflussen. Als Beispiel für den Einfluss, den die Hardware auf die Leistung hat, werden im Folgenden einige der Auswirkungen betrachtet, die mit dem Plattenspeicher zusammenhängen.

Vier Aspekte der Plattenspeicherverwaltung können sich auf die Leistung auswirken:

- **Speichereinteilung**

  Wie Sie eine begrenzte Speichergröße zwischen Indizes und Daten sowie unter Tabellenbereichen aufteilen, bestimmt in hohem Maße, welche Leistung die einzelnen Komponenten in verschiedenen Situationen erreichen.

- **Speicherverschwendung**

  Verschwendeter Speicher wirkt sich vielleicht als solcher nicht auf die Leistung des Systems aus, das ihn besitzt, aber er ist eine Ressource, die zur Verbesserung der Leistung an anderer Stelle genutzt werden könnte.
- **Verteilung der Platten-E/A**

  Wie ausgewogen Sie den Bedarf an Platten-E/A auf mehrere Plattenspeichereinheiten und Controller verteilen, kann sich auf die Geschwindigkeit auswirken, mit der der Datenbankmanager Informationen von Platten abrufen kann.
- **Mangel an verfügbarem Speicher**

  Bei Erreichen der Grenze des verfügbaren Speichers kann die allgemeine Leistung beeinträchtigt werden.

Zugehörige Konzepte:

- „Informationen zu DMS-Einheiten" auf Seite 301
- „Verzeichnisse und Dateien einer Datenbank" auf Seite 13
- „SMS-Tabellenbereiche" auf Seite 16
- „DMS-Tabellenbereiche" auf Seite 17
- „Tabellen- und Indexverwaltung für Standardtabellen" auf Seite 20
- „Tabellen- und Indexverwaltung für MDC-Tabellen" auf Seite 24

Verzeichnisse und Dateien einer Datenbank

Beim Erstellen einer Datenbank werden zugehörige Informationen einschließlich Standardinformationen in einer Verzeichnishierarchie gespeichert. Die hierarchische Verzeichnisstruktur wird an der Speicherposition erstellt, die durch die von Ihnen im Befehl CREATE DATABASE angegebenen Informationen festgelegt wird. Wenn Sie beim Erstellen der Datenbank die Position des Verzeichnispfads oder Laufwerks nicht angeben, wird die Standardposition verwendet. Es empfiehlt sich, explizit anzugeben, wo die Datenbank erstellt werden soll.

In dem Verzeichnis, das Sie im Befehl CREATE DATABASE angeben, wird ein Unterverzeichnis erstellt, das den Namen des erstellten Exemplars verwendet. Dieses Unterverzeichnis stellt sicher, dass Datenbanken, die in verschiedenen Exemplaren in demselben Verzeichnis erstellt werden, nicht denselben Pfad verwenden. Unterhalb des Unterverzeichnisses mit dem Exemplarnamen wird ein Unterverzeichnis mit dem Namen NODE0000 erstellt. Dieses Unterverzeichnis unterscheidet Partitionen in einer logisch partitionierten Datenbankumgebung. Unterhalb des Verzeichnisses mit dem NODE-Namen wird ein Unterverzeichnis mit dem Namen SQL00001 erstellt. Der Name dieses Unterverzeichnisses verwendet das Datenbanktoken und stellt die Datenbank dar, die erstellt wird. Das Verzeichnis SQL00001 enthält Objekte, die der ersten erstellten Datenbank zugeordnet sind. Nachfolgend erstellte Datenbanken erhalten höhere Nummern: SQL00002 usw. Diese Unterverzeichnisse unterscheiden Datenbanken, die in diesem Exemplar in dem Verzeichnis erstellt werden, das Sie im Befehl CREATE DATABASE angegeben haben.

Die Verzeichnisstruktur sieht wie folgt aus:

```
<ihr_verzeichnis>/<ihr_verzeichnis>/NODE0000/SQL00001/
```

Das Datenbankverzeichnis enthält die folgenden Dateien, die durch den Befehl CREATE DATABASE erstellt werden.

- Die Dateien SQLBP.1 und SQLBP.2 enthalten Pufferpoolinformationen. Jede Datei besitzt zur Sicherung eine Kopie.

- Die Dateien SQLSPCS.1 und SQLSPCS.2 enthalten Tabellenbereichsinformationen. Jede Datei besitzt zur Sicherung eine Kopie.
- Die Datei SQLDBCON enthält Datenbankkonfigurationsdaten. Editieren Sie diese Datei nicht. Verwenden Sie zum Ändern von Konfigurationsparametern entweder die Steuerzentrale oder die Befehlszeilenanweisungen UPDATE DATABASE CONFIGURATION und RESET DATABASE CONFIGURATION.
- Die Protokolldatei DB2RHIST.ASC und ihre Sicherungskopie DB2RHIST.BAK enthalten Protokollinformationen über Sicherungen, Wiederherstellungen, Ladeoperationen für Tabellen, Reorganisationen von Tabellen, Änderungen an Tabellenbereichen und andere Änderungen an einer Datenbank.

  Die Datei DB2TSCHNG.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 Wiederherstellung von Tabellenbereichen verwendet Informationen aus dieser Datei, um festzustellen, welche Protokolldateien bei einer Tabellenbereichswiederherstellung zu verarbeiten sind. Die können den Inhalt beider Protokolldateien in einem Texteditor untersuchen.
- Die Protokollsteuerdateien SQLOGCTL.LFH und SQLOGMIR.LFH enthalten Informationen zu den aktiven Protokollen.

  Bei der Wiederherstellungsverarbeitung wird anhand von Informationen aus dieser Datei festgestellt, an welcher Stelle in den Protokollen die Wiederherstellung beginnen soll. Das Unterverzeichnis SQLOGDIR enthält die eigentlichen Protokolldateien.

  Anmerkung: Sie sollten sicherstellen, dass das Unterverzeichnis für die Protokolle anderen Platten zugeordnet ist als denen, die für Ihre Daten verwendet werden. Ein Plattenproblem kann in diesem Fall auf Ihre Daten bzw. Ihre Protokolle beschränkt werden, so dass nicht beide gleichzeitig davon betroffen sind. Dies kann zudem einen deutlichen Vorteil für die Leistung mit sich bringen, da die Protokolldateien und die Datenbankbehälter nicht um die Bewegung derselben Plattenschreib-/leseköpfe konkurrieren. Ändern Sie die Position des Protokollunterverzeichnisses mit Hilfe des Konfigurationsparameters *newlogpath* der Datenbank.
- Die Datei SQLINSLK hilft bei der Sicherstellung, dass eine Datenbank nur von einem Exemplar des Datenbankmanagers verwendet wird.

Bei der Erstellung einer Datenbank wird gleichzeitig auch ein detaillierter Ereignismonitor für gegenseitige Sperren erstellt. Die Dateien des detaillierten Ereignismonitors für gegenseitige Sperren werden im Datenbankverzeichnis auf dem Katalogknoten 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 der Ausgabedateien, die nicht mehr benötigt werden, kann der Ereignismonitor bei der nächsten Datenbankaktivierung erneut aktiviert werden.

Weitere Informationen für SMS-Datenbankverzeichnisse

Die SQLT*-Unterverzeichnisse enthalten die SMS-Standardtabellenbereiche (System Managed Space, vom System verwalteter Speicher), die für eine betriebsfähige Datenbank erforderlich sind. Die folgenden drei Standardtabellenbereiche werden erstellt:

- 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 Behälter wird eine Datei mit dem Namen SQLTAG.NAM erstellt. Diese Datei markiert das betreffende Unterverzeichnis als in Gebrauch, so dass 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 Behälter 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-Tabelle Blockzuordnungsinformationen)
- SQL*.LF (enthält Daten der Typen LONG VARCHAR oder LONG VARGRAPHIC)
- SQL*.LB (enthält Daten der Typen BLOB, CLOB oder DBCLOB)
- 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)

Zugehörige Konzepte:

- „SMS- und DMS-Tabellenbereiche im Vergleich" in *Systemverwaltung: Konzept*
- „Informationen zu DMS-Einheiten" auf Seite 301
- „SMS-Tabellenbereiche" auf Seite 16
- „DMS-Tabellenbereiche" auf Seite 17
- „Illustration der Adressenzuordnung eines DMS-Tabellenbereichs" auf Seite 19
- „Datei des Wiederherstellungsprotokolls" in *Datenwiederherstellung und hohe Verfügbarkeit Handbuch und Referenz*

Zugehörige Referenzen:

- „CREATE DATABASE Command" in *Command Reference*

Übersicht über Tabellenbereiche

Die folgenden Abschnitte beschreiben die Tabellenbereiche und vergleichen die beiden Typen von Tabellenbereichen, die in DB2 verfügbar sind, im Hinblick auf ihre Vor- und Nachteile.

SMS-Tabellenbereiche

SMS-Tabellenbereiche (SMS - System Managed Space, vom System verwalteter Speicher) speichern Daten in Betriebssystemdateien. Die Daten in den Tabellenbereichen werden einheitenübergreifend in Speicherbereichen in allen Behältern des Systems gespeichert. Ein *Speicherbereich* ist eine Gruppe von aufeinander folgenden Seiten, die für die Datenbank definiert sind. Die Dateierweiterung bezeichnet den Datentyp, der in der betreffenden Datei gespeichert wird. Zur gleichmäßigen Datenverteilung auf alle Behälter im Tabellenbereich, werden die Anfangsspeicherbereiche für Tabellen reihum in allen Behältern angelegt. Eine solche Verteilung der Speicherbereiche ist besonders wichtig, wenn die Datenbank zahlreiche kleinen Tabellen enthält.

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 gesetzt wird, wird eine volle EXTENTSIZE-Größe zugeordnet, wenn Speicherbereich benötigt wird. Ansonsten wird Speicherbereich jedes Mal in der Größe von einer Seite zugeordnet. Vor Version 8.2 war die Standardeinstellung NO, so dass jedes Mal eine Seite zugeordnet wurde. Dieser Standardwert kann mit dem Tool *db2empfa* geändert werden. Wenn Sie db2empfa ausführen, wird der Datenbankkonfigurationsparameter *multipage_alloc* auf den Wert YES gesetzt. In Version 8.2 ist die Standardeinstellung des Konfigurationsparameters der Wert YES, so dass standardmäßig jedes Mal ein voller EXTENTSIZE großer Speicherbereich zugeordnet wird.

Die mehrseitige Dateizuordnung betrifft nur die Daten- und Indexteile einer Tabelle. Das heißt, die Dateien .LF, .LB und .LBA werden nicht jedes Mal um eine EXTENTSIZE-Größe erweitert.

Wenn der gesamte Speicherplatz in einem einzelnen Behälter in einem SMS-Tabellenbereich Tabellen zugeordnet ist, wird der Tabellenbereich als voll betrachtet, selbst wenn noch weiterer Speicherplatz in anderen Behältern verblieben ist. Sie können einem SMS-Tabellenbereich Behälter nur in einer Partition hinzufügen, die noch keine Behälter hat.

Anmerkung: SMS-Tabellenbereiche können die Vorteile des Vorablesezugriffs und der Cachefunktion des Dateisystems nutzen.

Zugehörige Konzepte:

- „Aufbau von Tabellenbereichen" in *Systemverwaltung: Konzept*
- „SMS- und DMS-Tabellenbereiche im Vergleich" in *Systemverwaltung: Konzept*

Zugehörige Tasks:

- „Hinzufügen eines Behälters in einem SMS-Tabellenbereich einer Partition" in *Systemverwaltung: Implementierung*

Zugehörige Referenzen:

- „multipage_alloc - Zuordnung aus mehreren Seiten bestehender Datei aktiv" auf Seite 500
- „db2empfa - Enable Multipage File Allocation Command" in *Command Reference*

DMS-Tabellenbereiche

Bei einem DMS-Tabellenbereich (DMS - Database-Managed Space, vom Datenbankmanager verwalteter Tabellenbereich) steuert der Datenbankmanager den Speicherbereich. Bei der Definition des DMS-Tabellenbereichs wird eine Liste mit Einheiten oder Dateien ausgewählt, die zu diesem gehören sollen. Der Speicherbereich in diesen Einheiten oder Dateien wird durch den DB2®-Datenbankmanager verwaltet. Ebenso wie SMS-Tabellenbereiche und SMS-Behälter verwenden auch DMS-Tabellenbereiche und der Datenbankmanager einheitenübergreifendes Lesen und Schreiben von Daten (engl. striping) für Speicherbereiche, um eine gleichmäßige Verteilung von Daten auf alle Behälter sicherzustellen.

DMS-Tabellenbereiche unterscheiden sich von SMS-Tabellenbereichen dadurch, dass bei DMS-Tabellenbereichen der Speicherbereich bereits bei der Erstellung des Tabellenbereichs zugeordnet wird und nicht erst, wenn er benötigt wird.

Darüber hinaus kann sich die Platzierung von Daten bei beiden Arten von Tabellenbereichen unterscheiden. Betrachten Sie beispielsweise den Bedarf an effizienten Tabellensuchoperationen: Es ist wichtig, dass die Seiten in einem Speicherbereich physisch aufeinander folgen. Bei SMS entscheidet das Dateisystem des Betriebssystems, wo die logischen Dateiseiten physisch abgelegt werden. Die Seiten können aufeinander folgend zugeordnet werden, müssen aber nicht. Dies hängt vom Umfang anderer Aktivitäten im Dateisystem sowie von dem Algorithmus ab, der zum Bestimmen der Platzierung verwendet wird. Bei DMS jedoch kann der Datenbankmanager sicherstellen, dass die Seiten physisch aufeinander folgen, da er direkt mit dem Plattendatenträger interagiert.

Anmerkung: Ebenso wie SMS-Tabellenbereiche können DMS-Dateibehälter die Vorteile des Vorablesezugriffs und der Cachefunktion des Dateisystems nutzen. Allerdings können DMS-Tabellenbereiche dies nicht.

Zu dieser allgemeinen Aussage bezüglich der aufeinander folgenden Platzierung von Seiten im Speicher gibt es eine Ausnahme. Beim Arbeiten mit DMS-Tabellenbereichen sind zwei Behälteroptionen verfügbar: unformatierte Einheiten (raw) und Dateien. Beim Arbeiten mit Dateibehältern ordnet der Datenbankmanager den gesamten Behälter bei der Erstellung des Tabellenbereichs zu. Ein Ergebnis dieser ersten Zuordnung des gesamten Tabellenbereichs besteht darin, dass die physische Zuordnung, zwar nicht garantiert, jedoch normalerweise direkt aufeinander folgend erfolgt, obwohl das Dateisystem die Zuordnung vornimmt. Beim Arbeiten mit Behältern aus unformatierten Einheiten übernimmt der Datenbankmanager die Steuerung der gesamten Einheit und stellt immer sicher, dass die Seiten in einem Speicherbereich direkt aufeinander folgen.

Im Unterschied zu SMS-Tabellenbereichen brauchen die Behälter, aus denen ein DMS-Tabellenbereich besteht, in Bezug auf ihre Kapazität nicht annähernd übereinzustimmen. Es empfiehlt sich jedoch, dass die Behälter in Bezug auf ihre Kapazität gleich oder annähernd gleich sind. Darüber hinaus kann in einem DMS-Tabellenbereich jeder verfügbare freie Speicherbereich anderer Behälter verwendet werden, wenn ein Behälter voll ist.

Beim Arbeiten mit DMS-Tabellenbereichen sollten Sie in Betracht ziehen, jeden Behälter 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 Behälter 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. Der EXTENTSIZE große Speicherbereich ist die Speicherzuordnungseinheit innerhalb eines Tabellenbereichs. Nur jeweils eine Tabelle oder ein anderes 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 SMP für den Tabellenbereich. SMP-Speicherbereiche sind in regelmäßigen Abständen über den gesamten Tabellenbereich verteilt. Jeder SMP-Speicherbereich besteht einfach aus einer Bitmaske der Speicherbereiche vom aktuellen SMP-Speicherbereich bis zum nächsten SMP-Speicherbereich. Die Bitmaske 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, Speicherbereichszuordnungsseite) befindet. Jedes Objekt verfügt über eigene EMPs, die eine Zuordnung zu allen Seiten des Objekts darstellen, das in der logischen Adressenzuordnung für den Speicherbereich gespeichert ist.

Zugehörige Konzepte:

- „Aufbau von Tabellenbereichen" in *Systemverwaltung: Konzept*
- „SMS- und DMS-Tabellenbereiche im Vergleich" in *Systemverwaltung: Konzept*
- „Informationen zu DMS-Einheiten" auf Seite 301
- „Verzeichnisse und Dateien einer Datenbank" auf Seite 13
- „SMS-Tabellenbereiche" auf Seite 16
- „Illustration der Adressenzuordnung eines DMS-Tabellenbereichs" auf Seite 19

Zugehörige Tasks:

- „Hinzufügen eines Behälters zu einem DMS-Tabellenbereich" in *Systemverwaltung: Implementierung*

Zugehörige Referenzen:

- „CREATE TABLESPACE statement" in *SQL Reference, Volume 2*

Illustration der Adressenzuordnung eines DMS-Tabellenbereichs

Die folgende Abbildung zeigt die logische Adressenzuordnung für einen DMS-Tabellenbereich.

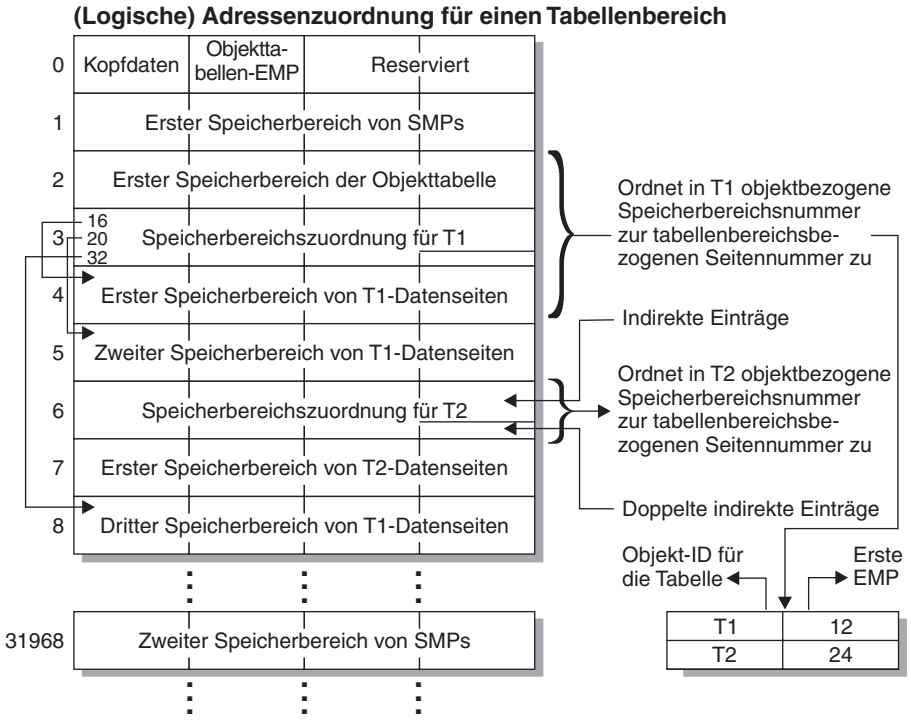


Abbildung 3. DMS-Tabellenbereiche

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 Behältern gespeichert, die dem Tabellenbereich zugeordnet sind.

Zugehörige Konzepte:

- „Informationen zu DMS-Einheiten" auf Seite 301
- „Leistungsfaktoren für Plattenspeichern" auf Seite 12
- „DMS-Tabellenbereiche" auf Seite 17

Tabellen und Indizes

Die folgenden Abschnitte behandeln die Verwaltung sowohl von Standardtabellen als auch von MDC-Tabellen sowie von Indizes für diese Tabellen.

Tabellen- und Indexverwaltung für Standardtabellen

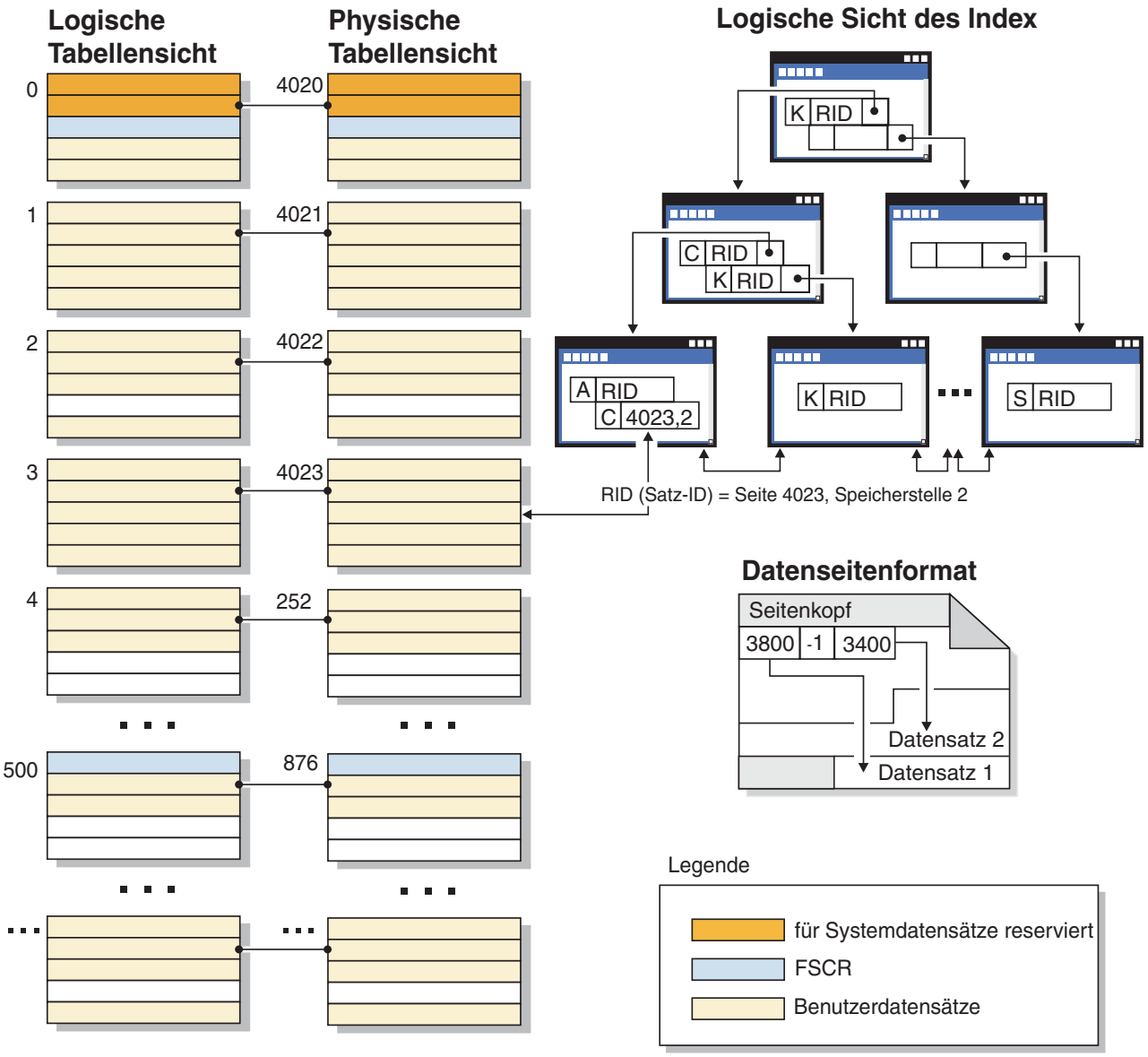


Abbildung 4. Logische Tabellen-, Datensatz und Indexstruktur für Standardtabellen

In Standardtabellen werden Tabellendaten logisch in Form einer Liste von Datenseiten organisiert. Diese Datenseiten werden logisch auf der Grundlage der EXTENTSIZE-Größe des Tabellenbereichs zu Gruppen zusammengefasst. Wenn die EXTENTSIZE-Größe beispielsweise vier beträgt, sind die Seiten null bis drei Teil des ersten EXTENTSIZE-Speicherbereichs, die Seiten vier bis sieben sind Teil des zweiten usw.

Die Zahl der Datensätze, die in den einzelnen Datenseiten enthalten sind, kann je nach Größe der Datenseite und Größe der Datensätze variieren. Maximal können 255 Datensätze auf einer Seite untergebracht werden. Die meisten Seiten enthalten nur Benutzerdatensätze. Eine kleine Zahl von Seiten enthält jedoch spezielle interne Datensätze, die von DB2® zur Verwaltung der Tabelle verwendet werden. Auf jeder 500sten Seite befindet sich beispielsweise ein FSCR-Datensatz (Free Space Control Record, Steuersatz für freien Speicherbereich). Diese Datensätze enthalten eine Zuordnung des freien Speicherbereichs für neue Datensätze auf jeder der folgenden 500 Datenseiten (bis zum nächsten FSCR-Datensatz). Dieser verfügbare freie Speicherbereich wird verwendet, wenn Datensätze in die Tabelle eingefügt werden.

Logisch werden Indexseiten als B-Baumstruktur organisiert, durch die Datensätze in der Tabelle, die einen angegebenen Schlüsselwert besitzen, auf effiziente Weise lokalisiert werden können. Die Zahl der Entitäten auf einer Indexseite ist nicht festgelegt, sondern hängt von der Größe des Schlüssels ab. Bei Tabellen in DMS-Tabellenbereichen verwenden RIDs (Record Identifiers, Satz-IDs) auf den Indexseiten Seitennummern, die nicht zum Objekt, sondern zum Tabellenbereich relativ sind. Dadurch kann bei einer **Indexsuche** direkt auf die Datenseiten zugegriffen werden, ohne dass eine Speicherbereichsmaske (EMP) für die Zuordnung erforderlich ist.

Jede Datenseite besitzt dasselbe Format. Alle Datenseiten beginnen mit einem Seitenkopf. Nach dem Seitenkopf folgt ein Speicherstellenverzeichnis. Jeder Eintrag im Speicherstellenverzeichnis entspricht einem anderen Datensatz auf der Seite. Der Eintrag selbst ist die relative Byteadresse auf der Datenseite, an der der Datensatz beginnt. Einträge mit minus eins (-1) entsprechen gelöschten Datensätzen.

Satz-IDs und Seiten

Satz-IDs (Record Identifier, RID) sind eine aus drei Byte bestehende Seitennummer gefolgt von einer Speicherstellennummer aus einem Byte. Datensätze von Indizes des Typs 2 enthalten ein zusätzliches Byte, das als ridFlag-Markierung bezeichnet wird. Die ridFlag-Markierung speichert Informationen über den Status von Schlüsseln im Index, zum Beispiel ob dieser Schlüssel als gelöscht markiert wurde. Wenn der Index zur Identifizierung einer Satz-ID verwendet wird, wird diese Satz-ID dazu verwendet, zur richtigen Datenseite und Speicherstellennummer auf dieser Seite zu gelangen. Wenn dem Datensatz eine Satz-ID zugeordnet ist, wird diese erst wieder bei einer Reorganisation der Tabelle geändert.

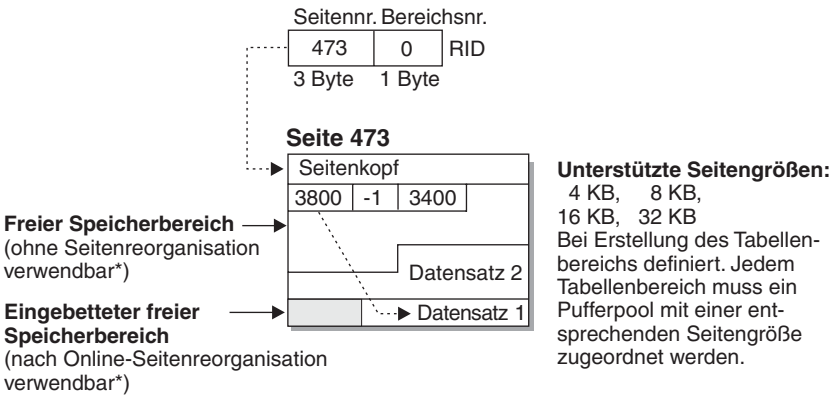


Abbildung 5. Format der Datenseite und der Satz-ID (RID)

Wenn eine Tabellenseite reorganisiert wird, wird eingebetteter freier Speicherbereich, der nach dem physischen Löschen eines Datensatzes auf der Seite verbleibt, in verwendbaren freien Speicherbereich umgewandelt. Satz-IDs werden durch Verschieben von Datensätzen auf einer Datenseite erneut definiert, damit der verwendbare freie Speicherbereich genutzt werden kann.

DB2 unterstützt verschiedene Seitengrößen. Verwenden Sie größere Seiten für Auslastungen, bei denen auf Zeilen normalerweise sequenziell zugegriffen wird. Sequenzieller Zugriff wird beispielweise für Anwendungen zur Entscheidungshilfe oder in Fällen verwendet, in denen ein starker Gebrauch von temporären Tabellen gemacht wird. Verwenden Sie geringere Seitengrößen für Auslastungen, deren Zugriff normalerweise wahlfrei erfolgt. Wahlfreier Zugriff wird beispielsweise in OLTP-Umgebungen verwendet.

Indexverwaltung in Standardtabellen

DB2-Indizes verwenden eine optimierte B-Baumstrukurimplementierung auf der Basis einer effizienten Indexverwaltungsmethode mit einem hohen Grad des gemeinsamen Zugriffs und einer im Voraus schreibenden Protokollierung (Write Ahead Logging).

Die optimierte B-Baumstrukturimplementierung verfügt über bidirektionale Zeiger auf den Blattseiten, mit denen ein einzelner Index sowohl vorwärts als auch rückwärts gerichtete Suchoperationen unterstützen kann. Indexseiten werden normalerweise in der Mitte geteilt, wobei die HIGHKEY-Seite eine Ausnahme bildet, bei der eine 90/10-Teilung erfolgt. Das heißt, dass die hohen zehn Prozent der Indexschlüssel auf der neuen Seite Platz finden. Diese Art der Indexseitenteilung ist bei Auslastungen nützlich, bei denen oft INSERT-Anforderungen mit neuen HIGHKEY-Werten durchgeführt werden.

Seit Version 8.1 arbeitet DB2 mit Indizes des Typs 2. Wenn Sie eine Migration von früheren Versionen von DB2 durchführen, werden sowohl Indizes des Typs 1 als auch Indizes des Typs 2 verwendet, bis Sie die Indizes reorganisieren oder andere Aktionen durchführen, die Indizes des Typs 1 in Indizes des Typs 2 konvertieren.

Der Indextyp bestimmt, wie gelöschte Schlüssel physisch von den Indexseiten entfernt werden.

- Für Indizes des Typs 1 werden Schlüssel von den Indexseiten beim Löschen der Schlüssel entfernt und Indexseiten freigegeben, wenn der letzte Indexschlüssel von der Seite entfernt ist.
- Für Indizes des Typs 2 werden Indexschlüssel beim Löschen des Schlüssels nur dann von der Seite entfernt, wenn eine exklusive Sperre (Modus X) für die Tabelle aktiviert ist. Wenn Schlüssel nicht sofort entfernt werden können, werden sie als gelöscht markiert und später entfernt. Weitere Informationen enthält der Abschnitt zu Indizes des Typs 2.

Wenn Sie die Online-Indexdefragmentierung aktiviert haben, indem Sie bei der Erstellung des Index die Klausel MINPCTUSED auf einen höheren Wert als null gesetzt haben, können Blattseiten online zusammengefügt werden. Der von Ihnen angegebene Wert ist der Schwellenwert für den minimalen Prozentsatz an Speicherplatz, der auf den Indexblattseiten genutzt wird. Wenn ein Schlüssel von einer Indexseite entfernt wurde und der Prozentsatz an Speicherplatz auf der Seite bei oder unter dem angegebenen Wert liegt, versucht der Datenbankmanager die verbleibenden Schlüssel mit denen einer benachbarten Seite zu einer Seite zusammenzufügen. Wenn genügend Platz vorhanden ist, wird die Operation zum Zusammenfügen ausgeführt und eine Indexseite gelöscht. Die Online-Indexdefragmentierung kann die Wiederverwendung von Speicherbereich verbessern. Wenn jedoch der Wert für MINPCTUSED zu hoch ist, wird mehr Zeit auf Versuche verwendet, Seiten zusammenzufügen, die jedoch mit geringerer Wahrscheinlichkeit erfolgreich sind. Der empfohlene Wert für diese Klausel ist 50 % oder weniger.

Anmerkung: Da die Onlinedefragmentierung nur stattfindet, wenn Schlüssel von einer Indexseite entfernt werden, findet sie in einem Index des Typs 2 nicht statt, wenn Schlüssel nur als gelöscht markiert, jedoch nicht physisch von der Seite entfernt wurden.

Die Klausel INCLUDE der Anweisung CREATE INDEX ermöglicht das Einfügen einer angegebenen Spalte bzw. von Spalten auf den Indexseiten zusätzlich zu den Schlüsselspalten. Dadurch kann die Zahl der Abfragen steigen, bei denen ein reiner Indexzugriff möglich ist. Gleichzeitig können jedoch die Anforderungen an Indexspeicherplatz und möglicherweise auch der Verwaltungsaufwand für den betreffenden Index steigen, wenn die eingefügten Spalten häufig aktualisiert werden. Der Verwaltungsaufwand zur Aktualisierung von INCLUDE-Spalten ist geringer als der zur Aktualisierung von Schlüsselspalten, jedoch höher als der zur Aktualisierung von Spalten, die nicht im Index vertreten sind. Das Ordnen der Index-B-Baumstruktur geschieht nur unter Verwendung der Schlüsselspalten, nicht der eingeschlossenen Spalten.

Zugehörige Konzepte:

- „Speicherbedarf für Datenbankobjekte" in *Systemverwaltung: Konzept*
- „Überlegungen zur Auswahl von Tabellenbereichen für Tabellen" in *Systemverwaltung: Konzept*
- „Entwerfen von Tabellen mit mehrdimensionalem Clustering (MDC)" in *Systemverwaltung: Konzept*
- „Tabellen- und Indexverwaltung für MDC-Tabellen" auf Seite 24
- „Erstellung, Platzierung und Verwendung von Tabellen mit mehrdimensionalem Clustering (MDC)" in *Systemverwaltung: Konzept*
- „Indexbereinigung und Indexpflege" auf Seite 296
- „Einfügeverarbeitung" auf Seite 30

Tabellen- und Indexverwaltung für MDC-Tabellen

Die Tabellen- und Indexorganisation für Tabellen mit mehrdimensionalem Clustering (MDC) basiert auf den gleichen logischen Strukturen wie die Organisation von Standardtabellen. Ebenso wie Standardtabellen werden MDC-Tabellen in Seiten organisiert, die Datenzeilen enthalten, die wiederum in Spalten unterteilt sind. Die Zeilen jeder Seite werden durch Zeilen-IDs (RIDs) gekennzeichnet. Darüber hinaus werden die Seiten von MDC-Tabellen in EXTENTSIZE große Blöcke gruppiert. In der folgenden Abbildung, die eine Tabelle mit dem EXTENTSIZE-Wert 4 darstellt, bilden die ersten vier Seiten mit den Nummern 0 bis 3 den ersten Block in der Tabelle. Die nächste Gruppe von Seiten mit den Nummern 4 bis 7 bildet den zweiten Block der Tabelle.

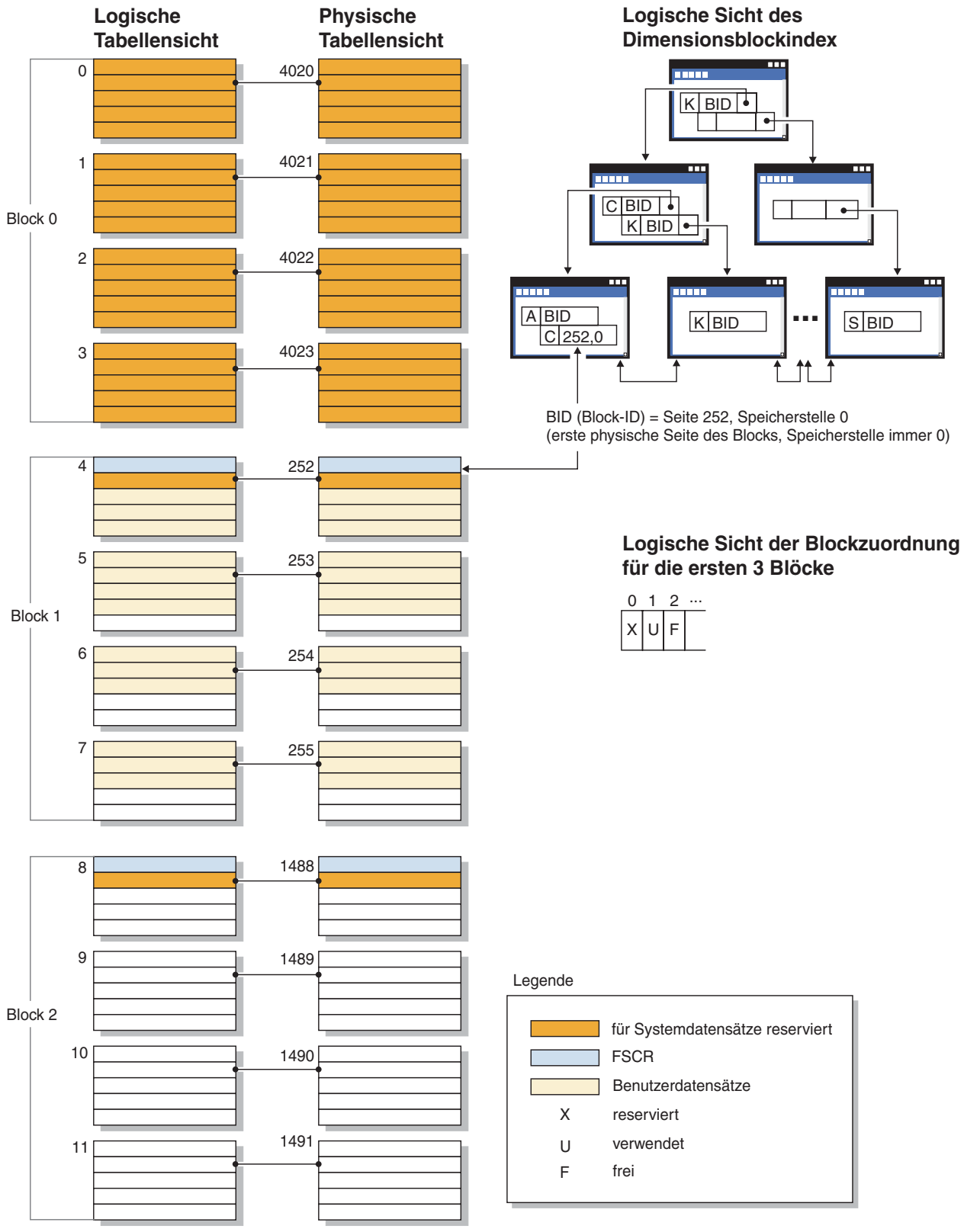


Abbildung 6. Logische Tabellen-, Datensatz- und Indexstruktur für MDC-Tabellen

Der Block enthält spezielle interne Datensätze, die von DB2® zur Verwaltung der Tabelle verwendet werden. Dazu gehören auch die FSCR-Sätze, d. h. die Steuerdatensätze für freie Speicherbereiche (FSCR - Free-Space Control Record). In den

nachfolgenden Blöcken enthält jeweils die erste Seite die FSCR-Sätze. Ein FSCR-Satz ordnet den freien Speicher für neue Datensätze zu, der auf den jeweiligen Seiten im Block vorhanden ist. Dieser verfügbare freie Speicherbereich wird verwendet, wenn Datensätze in die Tabelle eingefügt werden.

Wie am Namen ersichtlich, ordnen MDC-Tabellen ihre Daten in mehr als einer Dimension in so genannten Clustern (d. h. in Datengruppen) an. Jede Dimension wird durch eine Spalte bzw. eine Gruppe von Spalten bestimmt, die Sie in der Klausel ORGANIZE BY DIMENSIONS der Anweisung CREATE TABLE angeben. Bei der Erstellung einer MDC-Tabelle werden die beiden folgenden Arten von Indizes automatisch erstellt:

- Ein Dimensionsblockindex, der Zeiger auf jeden belegten Block für eine einzelne Dimension enthält.
- Einen zusammengesetzten Blockindex, der alle Dimensionsschlüsselspalten enthält. Der zusammengesetzte Blockindex dient zur Erhaltung der Clusterbildung bei Einfüge- und Aktualisierungsaktivitäten.

Das Optimierungsprogramm zieht Zugriffspläne unter Nutzung von Dimensionsblockindizes in Betracht, wenn es den effizientesten Zugriffsplan für eine bestimmte Abfrage ermittelt. Wenn Abfragen Vergleichselemente für Dimensionswerte enthalten, kann das Optimierungsprogramm den Dimensionsblockindex verwenden, um die (EXTENTSIZE großen) Speicherbereiche, die diese Werte enthalten, zu ermitteln und aus diesen Daten abzurufen. Da diese Speicherbereiche physisch aufeinander folgende Seiten auf der Platte darstellen, führt dieses Verfahren zu einer effizienteren Leistung und minimiert den E/A-Aufwand.

Darüber hinaus können Sie auch spezielle Zeilen-ID-Indizes (RID-Indizes) erstellen, wenn die Analyse der Datenzugriffspläne darauf hinweist, dass solche Indizes die Abfrageleistung verbessern würden.

Neben den Dimensionsblockindizes und dem zusammengesetzten Blockindex pflegen MDC-Tabellen eine Blockzuordnung (engl. block map), die eine Bitmaske enthält, die den Verfügbarkeitsstatus jedes Blocks angibt. Die folgenden Attribute werden in der Liste der Bitmaske codiert:

- X (reserviert): Der erste Block enthält nur Systeminformationen für die Tabelle.
- U (in Gebrauch): Dieser Block wird verwendet und ist einem Dimensionsblockindex zugeordnet.
- L (geladen): In diesen Block wurden durch eine aktuelle Ladeoperation Daten geladen.
- C (Integritätsbedingung prüfen): Dieser Block wurde durch die Ladeoperation so eingestellt, dass er eine inkrementelle Prüfung auf Integritätsbedingungen während des Ladens angibt.
- T (Tabelle aktualisieren): Dieser Block wurde durch die Ladeoperation so eingestellt, dass er eine erforderliche Pflege von AST-Tabellen angibt.
- F (frei): Wenn kein anderes Attribut gesetzt ist, wird der Block als frei betrachtet.

Da jeder Block einen Eintrag in der Blockzuordnungsdatei hat, wächst die Datei, wenn die Tabelle größer wird. Diese Datei wird als separates Objekt gespeichert. In einem SMS-Tabellenbereich hat sie einen neuen Dateityp. In einem DMS-Tabellenbereich hat sie einen neuen Objektdeskriptor in der Objekttabelle.

Zugehörige Konzepte:

- „Speicherbedarf für Datenbankobjekte" in *Systemverwaltung: Konzept*

- „Entwerfen von Tabellen mit mehrdimensionalem Clustering (MDC)" in *Systemverwaltung: Konzept*
- „Erstellung, Platzierung und Verwendung von Tabellen mit mehrdimensionalem Clustering (MDC)" in *Systemverwaltung: Konzept*
- „Einfügeverarbeitung" auf Seite 30

Indexstruktur

Der Datenbankmanager verwendet eine B+-Baumstruktur zur Indexspeicherung. Eine B+-Baumstruktur hat eine oder mehrere Stufen, wie die folgende Abbildung zeigt (`rid` steht für Satz/Zeilen-ID):

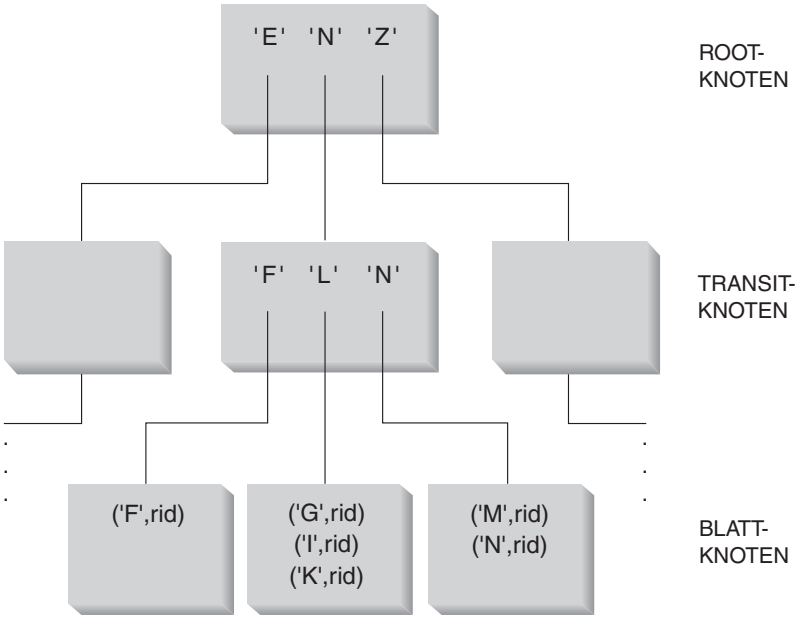


Abbildung 7. B+-Baumstruktur

Die oberste Stufe wird als *Rootknoten* bezeichnet. Die unterste Stufe besteht aus *Blattknoten*, in denen die Werte des Indexschlüssels jeweils mit Zeigern auf die Zeile in der Tabelle gespeichert sind, die den Schlüsselwert enthält. Die Stufen zwischen dem Rootknoten und dem Blattknoten werden als *Transitknoten* bezeichnet.

Bei der Suche nach einem bestimmten Indexschlüsselwert durchsucht der Indexmanager den Indexbaum ausgehend vom Rootknoten. Der Rootknoten enthält einen Schlüssel für jeden Knoten auf der folgenden Stufe. Der Wert jedes dieser Schlüssel ist jeweils der größte vorhandene Schlüsselwert für den entsprechenden Knoten auf der nächsten Stufe. Wenn zum Beispiel ein Index drei Stufen hat, wie in der Abbildung gezeigt, dann durchsucht der Indexmanager zum Auffinden eines Indexschlüsselwerts den Rootknoten nach dem ersten Schlüsselwert, der größer oder gleich dem gesuchten Schlüsselwert ist. Der Rootknotenschlüssel enthält dann einen Zeiger auf einen bestimmten Transitknoten. Der Indexmanager setzt dieses Verfahren durch die Transitknoten fort, bis er den Blattknoten findet, der den benötigten Indexschlüssel enthält.

In der Abbildung wird zum Beispiel nach dem Schlüssel „I" gesucht. Der erste Schlüssel im Rootknoten, der größer oder gleich „I" ist, ist „N". Dieser Wert zeigt auf den mittleren Knoten auf der nächsten Stufe. Der erste Schlüssel in diesem Transitknoten, der größer als oder gleich „I" ist, ist der Wert „L". Dieser Wert zeigt

auf einen bestimmten Blattknoten, in dem der Indexschlüssel für „I" zusammen mit der entsprechenden Zeilen-ID (RID) gefunden wird. Die Zeilen-ID (oder Satz-ID) gibt die entsprechende Zeile in der Basistabelle an. Die Blattknotenstufe kann auch Zeiger auf frühere Blattknoten enthalten. Diese Zeiger geben dem Indexmanager die Möglichkeit, die Blattknoten in beide Richtungen zu durchsuchen, um einen Bereich von Werten abzurufen, nachdem er einen Wert des Bereichs gefunden hat. Die Möglichkeit, den Index in beide Richtungen zu durchsuchen, ist nur gegeben, wenn der Index mit der Klausel ALLOW REVERSE SCANS erstellt wurde.

Für MDC-Tabellen (MDC - Mehrdimensionales Clustering) wird automatisch ein Blockindex für jede Clusteringdimension erstellt, die Sie für die Tabelle angeben. Außerdem wird ein zusätzlicher zusammengesetzter Blockindex erstellt, der einen Schlüsselbestandteil für jede Spalte enthält, die an einer Dimension der Tabelle beteiligt ist. Solche Indizes enthalten Zeiger auf Block-IDs (BIDs) anstelle von Satz-IDs (RIDs) und bieten Verbesserungen beim Datenzugriff.

In DB2® Version 8.1 und späteren Versionen können Indizes den Typ 1 oder den Typ 2 besitzen. Ein *Index des Typs 1* ist die ältere Art von Index. Indizes, die in früheren Versionen von DB2 erstellt wurden, sind vom Typ 1.

Ein Index des Typs 2 ist etwas größer als ein Index des Typs 1 und verfügt über Zusatzfunktionen, die das Sperren des jeweils nächsten Schlüssels minimieren. Mit der ein Byte großen *ridFlag*-Markierung, die mit jeder Satz-ID auf der Blattseite eines Index des Typs 2 gespeichert wird, wird eine Satz-ID ggf. als logisch gelöscht markiert, so dass sie später physisch gelöscht werden kann. Für jede Spalte variabler Länge, die im Index enthalten ist, wird in einem zusätzlichen Byte die tatsächliche Länge des Spaltenwerts gespeichert. Indizes des Typs 2 können auch deshalb größer als Indizes des Typs 1 sein, weil einige Schlüssel als gelöscht markiert, jedoch noch nicht von der Indexseite physisch gelöscht wurden. Nach der Festschreibung der DELETE- oder UPDATE-Transaktion können die als gelöscht markierten Schlüssel bereinigt werden.

Zugehörige Konzepte:

- „Vor- und Nachteile von Indizes" auf Seite 288
- „Indexreorganisation" auf Seite 298
- „Online-Indexdefragmentierung" auf Seite 300

Prozesse

Die folgenden Abschnitte enthalten allgemeine Beschreibungen der DB2-Prozesse.

Protokollverarbeitung

Alle Datenbanken pflegen Protokolldateien, die Datensätze über Datenbankänderungen aufzeichnen. Zwei Protokollierungsstrategien stehen zur Auswahl:

- Die Umlaufprotokollierung, bei der die Protokollsätze die Protokolldateien vollständig füllen und anschließend die ersten Protokollsätze in der ersten Protokolldatei überschreiben. Die überschriebenen Protokollsätze sind nicht wiederherstellbar.
- Behalten und Speichern der Protokollsätze, bei dem eine Protokolldatei archiviert wird, wenn sie mit Protokollsätzen gefüllt ist. Für weitere Protokollsätze werden neue Protokolldateien verfügbar gemacht.

Die Speicherung der Protokolldateien ermöglicht eine **aktualisierende Wiederherstellung**.Die aktualisierende Wiederherstellung wendet Änderungen an der Datenbank auf der Basis der abgeschlossenen und im Protokoll aufgezeichneten Arbeitseinheiten (Transaktionen) erneut an. Sie können angeben, dass die aktualisierende Wiederherstellung bis zum Ende der Protokolle oder bis zu einem bestimmten Zeitpunkt vor dem Ende der Protokolle durchgeführt wird.

Unabhängig von der Protokollierungsstrategie werden alle Änderungen an regulären Daten- und Indexseiten in den Protokollpuffer geschrieben. Die Daten im Protokollpuffer werden durch den Protokollprozess auf die Platte geschrieben. Unter folgenden Umständen muss die Abfrageverarbeitung darauf warten, dass Protokolldaten auf die Platte geschrieben werden:

- Bei einer COMMIT-Operation.
- Bevor die entsprechenden Datenseiten auf Platte geschrieben werden, da DB2® mit einer Vorabschreibprotokollierung (write ahead) arbeitet. Ein Vorteil der Vorabschreibprotokollierung besteht darin, dass bei Beendigung einer Transaktion durch Ausführen einer COMMIT-Anweisung nicht alle geänderten Daten und Indexseiten auf Platte geschrieben werden müssen.
- Bevor einige Änderungen an Metadaten vorgenommen werden, von denen sich die meisten aus der Ausführung von DDL-Anweisungen ergeben.
- Beim Schreiben von Protokollsätzen in den Protokollpuffer, wenn der Protokollpuffer voll ist.

DB2 verwaltet das Schreiben von Protokolldaten auf Platte in dieser Weise, um die Verarbeitungsverzögerung zu minimieren. In einer Umgebung, in der viele kurze gleichzeitige Transaktionen stattfinden, wird der größte Teil der Verarbeitungsverzögerung durch COMMIT-Anweisungen verursacht, die darauf warten müssen, dass Protokolldaten auf Platte geschrieben werden. Infolgedessen schreibt der Protokollprozess häufig kleine Mengen von Protokolldaten auf Platte, wobei eine zusätzliche Verzögerung durch den Systemaufwand für Protokoll-E/A entsteht. Um die Anwendungsantwortzeit gegen eine solche Protokollierungsverzögerung zu verbessern, setzen Sie den Datenbankkonfigurationsparameter *mincommit* auf einen höheren Wert als 1. Diese Einstellung kann zwar längere Verzögerungen für COMMIT-Operationen aus einigen Anwendungen verursachen, jedoch können mehr Protokolldaten in einer Operation auf Platte geschrieben werden.

Änderungen an großen Objekten (LOBs) und LONG VARCHAR-Daten werden mit Hilfe des Erstellens einer Spiegelkopie für Speicherseiten (Shadow Paging) protokolliert. Änderungen an LOB-Spalten werden nur dann protokolliert, wenn Sie die Beibehaltung der Protokolldateien (log_retain) angeben und die LOB-Spalte in der Anweisung CREATE TABLE ohne die Klausel NOT LOGGED definiert wurde. Änderungen an den Zuordnungsseiten für LONG- oder LOB-Datentypen werden wie reguläre Datenseiten protokolliert.

Zugehörige Konzepte:

- „Aktualisierungsverarbeitung” auf Seite 31
- „Client-/Serververarbeitungsmodell” auf Seite 32

Zugehörige Referenzen:

- „mincommit - Anzahl der Gruppenfestschreibungen” auf Seite 471

Einfügeverarbeitung

Wenn SQL-Anweisungen mit Hilfe von INSERT neue Informationen in eine Tabelle einfügen, durch ein INSERT-Suchalgorithmus zunächst die FSCR-Sätze (Free Space Control Records), um eine Seite mit genügend Speicherplatz zu finden. Selbst wenn der FSCR jedoch angibt, dass auf einer bestimmten Seite genügend freier Speicherbereich vorhanden ist, kann dieser jedoch möglicherweise nicht verwendet werden, da er durch eine nicht festgeschriebene Anweisung DELETE von einer anderen Transaktion reserviert ist. Um sicherzustellen, dass nicht festgeschriebener freier Speicherplatz verwendbar ist, sollten Sie Transaktionen häufig durch COMMIT festschreiben. Die Einstellung der Registriervariablen DB2MAXFSCRSEARCH bestimmt die Anzahl von FSCRs in einer Tabelle, die für eine INSERT-Anweisung durchsucht werden. Der Standardwert für diese Registriervariable ist fünf. Wenn in der angegebenen Anzahl von FSCRs kein Speicherplatz gefunden wird, wird der einzufügende Datensatz an das Ende der Tabelle angehängt. Zur Optimierung der Geschwindigkeit der Anweisung INSERT werden nachfolgende Datensätze ebenfalls an das Ende der Tabelle angehängt, bis zwei EXTENTSIZE-Speicherbereiche gefüllt sind. Wenn diese beiden Speicherbereiche gefüllt sind, setzt die nächste Anweisung INSERT die Suche bei dem FSCR fort, bei dem die letzte Suche beendet wurde.

Anmerkung: Setzen Sie die Registriervariable DB2MAXFSCRSEARCH auf einen kleinen Wert, um die INSERT-Geschwindigkeit, allerdings mit der möglichen Folge eines schnelleren Anwachsens von Tabellen, zu optimieren. Zur Optimierung der Wiederverwendung von Speicherplatz, möglicherweise zu Lasten der INSERT-Geschwindigkeit, setzen Sie die Registriervariable DB2MAXFSCRSEARCH auf einen höheren Wert.

Wenn alle FSCRs in der gesamten Tabelle auf diese Weise durchsucht wurden, werden die einzufügenden Datensätze ohne weiteres Suchen an das Ende angehängt. Es findet kein weiteres Suchen mit Hilfe der FSCRs mehr statt, sofern nicht Speicherplatz an einer Stelle in der Tabelle frei wird, wie zum Beispiel nach einer DELETE-Operation.

Es gibt zwei weitere INSERT-Algorithmusoptionen:

- Anhängemodus (APPEND MODE)

  In diesem Modus werden neue Zeilen immer an das Ende der Tabelle angehängt. Es finden weder Such- noch Pflegeoperationen für FSCRs statt. Diese Option wird mit der Anweisung ALTER TABLE APPEND ON aktiviert und kann die Leistung für ausschließlich wachsende Tabellen wie Journale verbessern.
- Ein Clusterindex ist für die Tabelle definiert.

  In diesem Fall versucht der Datenbankmanager, Datensätze auf derselben Seite wie andere Datensätze mit ähnliche Indexschlüsselwerten einzufügen. Wenn auf der betreffenden Seite kein Speicherbereich verfügbar ist, wird versucht, den Datensatz auf den umgebenden Seiten unterzubringen. Wenn dies keinen Erfolg hat, wird der oben beschriebene FSCR-Suchalgorithmus verwendet, jedoch mit dem Unterschied, dass der am schlechtesten passende Speicherplatz, nicht der erste passende gesucht wird. Durch dieses Verfahren werden normalerweise Seiten ausgewählt, die mehr freien Speicherbereich enthalten. Diese Methode richtet einen neuen Clustering-Bereich für Zeilen mit diesem Schlüsselwert ein.

  Wenn Sie einen Clusterindex für eine Tabelle definieren, verwenden Sie die Anweisung ALTER TABLE... PCTFREE, bevor Sie Daten in die Tabelle laden

oder die Tabelle reorganisieren. Die Klausel PCTFREE definiert einen Prozentsatz an freiem Speicherplatz, der auf der Datenseite der Tabelle nach dem Laden und Reorganisieren verbleiben soll. Dadurch steigt die Wahrscheinlichkeit, dass bei der Clusterindexoperation auf der richtigen Seite freier Speicherplatz gefunden wird.

Zugehörige Konzepte:

- „Tabellen- und Indexverwaltung für Standardtabellen" auf Seite 20
- „Aktualisierungsverarbeitung" auf Seite 31
- „Tabellen- und Indexverwaltung für MDC-Tabellen" auf Seite 24

Aktualisierungsverarbeitung

Wenn ein Agent eine Seite aktualisiert, arbeitet der Datenbankmanager mit dem folgenden Protokoll, um die für die Transaktion erforderliche E/A zu minimieren und die Wiederherstellbarkeit zu gewährleisten.

1. Die zu aktualisierende Seite wird fixiert und mit einer exklusiven Sperre belegt. In den Protokollpuffer wird ein Protokollsatz geschrieben, der beschreibt, wie die Änderung wiederholt und widerrufen werden kann. Während dieser Aktion wird auch eine Protokollfolgenummer (Log Sequence Number, LSN) empfangen und in den Kopfdaten der zu aktualisierenden Seite gespeichert.
2. Die Änderung wird an der Seite vorgenommen.
3. Die Sperre und die Fixierung der Seite werden aufgehoben.

   Die Seite gilt als „benutzt", da Änderungen an der Seite noch nicht auf Platte gespeichert wurden.
4. Der Protokollpuffer wird aktualisiert.

   Sowohl die Daten im Protokollpuffer als auch die „benutzte" Datenseite werden zwangsweise auf Platte geschrieben.

Zur Erzielung einer besseren Leistung werden diese E/A-Operationen verzögert, bis ein geeigneter Zeitpunkt eintritt, zum Beispiel eine Phase mit niedriger Systemauslastung, bis sie erforderlich sind, um die Wiederherstellbarkeit sicherzustellen, oder um die Wiederherstellungsdauer zu begrenzen. Speziell wird eine „benutzte" Seite zu folgenden Zeitpunkten gezwungenermaßen auf Platte geschrieben:

- Wenn ein anderer Agent sie auswählt.
- Wenn eine Seitenlöschfunktion die Seite aus einem der folgenden Gründe bearbeitet:
  - Ein anderer Agent wählt sie aus.
  - Der durch den Datenbankkonfigurationsparameter *chngpgs_thresh* angegebene Prozentsatz wird überschritten. Wenn dieser Wert überschritten wird, werden asynchrone Seitenlöschfunktionen aktiv und schreiben geänderte Seiten auf Platte. Wenn die proaktive Seitenbereinigung aktiviert ist, ist dieser Wert irrelevant und löst keine Seitenlöschfunktionen aus.
  - Der durch den Datenbankkonfigurationsparameter *softmax* angegebene Prozentsatz wird überschritten. Wenn dieser Wert überschritten ist, werden asynchrone Seitenlöschfunktionen aktiv und schreiben geänderte Seiten auf Platte. Wenn die proaktive Seitenbereinigung für die Datenbank aktiviert und die Anzahl von Seitenlöschfunktionen für die Datenbank richtig konfiguriert ist, sollte dieser Wert nie überschritten werden.
  - Die Anzahl sauberer Seiten auf der Vermeidungsliste fällt auf einen zu niedrigen Wert. Seitenlöschfunktionen reagieren auf diese Bedingung nur unter der Methode der proaktiven Seitenbereinigung.
  - Wenn benutzte Seiten zurzeit oder absehbar zu einer LSNGAP-Bedingung führen. Seitenlöschfunktionen reagieren auf diese Bedingung nur unter der Methode der proaktiven Seitenbereinigung.

- Wenn die Seite als Teil einer Tabelle aktualisiert wurde, für die die Klausel NOT LOGGED INITIALLY aufgerufen wird, und eine COMMIT-Anweisung abgesetzt wird. Wenn die COMMIT-Anweisung ausgeführt wird, werden alle geänderten Seiten auf Platte geschrieben, um die Wiederherstellbarkeit sicherzustellen.

Zugehörige Konzepte:

- „Protokollverarbeitung" auf Seite 28
- „Client-/Serververarbeitungsmodell" auf Seite 32

Zugehörige Referenzen:

- „softmax - Vor bedingtem Prüfpunkt zu schreibende Protokollsätze" auf Seite 474
- „chngpgs_thresh - Schwellenwert für geänderte Seiten" auf Seite 435

Client-/Serververarbeitungsmodell

Lokale und ferne Anwendungsprozesse können mit derselben Datenbank arbeiten. Eine ferne Anwendung initiiert eine Datenbankaktion von einer Maschine aus, die sich von der Datenbankmaschine entfernt befindet. Lokale Anwendungen sind auf der Servermaschine direkt mit der Datenbank verbunden.

Anmerkung: Wie DB2® Clientverbindungen verwaltet, hängt davon ab, ob der Verbindungskonzentrator aktiviert ist oder nicht. Der Verbindungskonzentrator ist aktiviert, wenn der Konfigurationsparameter *max_connections* des Datenbankmanagers auf einen höheren Wert als der Konfigurationsparameter *max_coordagents* gesetzt ist.

- Wenn der Verbindungskonzentrator inaktiviert ist, wird jeder Clientanwendung eine eindeutige EDU (Engine Dispatchable Unit) zugeordnet, die als *Koordinatoragent* bezeichnet wird und die die Verarbeitung für diese Anwendung koordiniert und mit der Anwendung kommuniziert.
- Wenn der Verbindungskonzentrator aktiviert ist, kann jeder Koordinatoragent viele Clientverbindungen jeweils einzeln verwalten und ggf. die anderen Arbeitsagenten zur Erledigung dieser Arbeit koordinieren. Für Internetanwendungen mit vielen relativ kurzfristigen Verbindungen oder ähnliche Anwendungen mit zahlreichen relativ kleinen Transaktionen verbessert der Verbindungskonzentrator die Leistung, indem er wesentlich mehr Clientanwendungen die Herstellung einer Verbindung ermöglicht. Darüber hinaus verringert er die Belastung der Systemressourcen durch jede einzelne Verbindung.

Alle Kreise in der folgenden Abbildung stellen EDUs (Engine Dispatchable Units, zuteilbare Einheiten der Steuerkomponente) dar, die auf UNIX®-Plattformen als „Prozesse" und unter Windows NT als „Threads" bezeichnet werden. Bevor die Arbeit ausgeführt werden kann, die für die Anwendung in der Datenbank durchgeführt werden soll, muss zwischen einer Anwendung und dem Datenbankmanager ein Kommunikationsmittel eingerichtet werden.

Bei A1 in der folgenden Abbildung richtet ein lokaler Client eine Kommunikationsverbindung zunächst über eine db2ipccm-EDU für ein. Bei A2 arbeitet die db2ipccm-EDU mit einer db2agent-EDU, die zum Koordinatoragenten für die Anwendungsanforderungen vom lokalen Client wird. Der Koordinatoragent nimmt anschließend bei A3 mit der Clientanwendung Kontakt auf, um zwischen der Clientanwendung und dem Koordinator eine Kommunikation über gemeinsamen Speicher einzurichten. Die Anwendung auf dem lokalen Client wird bei A4 mit der Datenbank verbunden.

Bei B1 in der folgenden Abbildung stellt ein ferner Client eine Kommunikationsverbindung über eine db2tcpcm-EDU her. Wenn ein anderes Kommunikationsprotokoll ausgewählt wird, wird der entsprechende Kommunikationsmanager verwendet. Die db2tcpcm-EDU richtet eine TCP/IP-Kommunikation zwischen der Clientanwendung und der db2tcpcm-EDU ein. Dann arbeitet sie mit einer db2agent-EDU bei B2, die zum Koordinatoragenten für die Anwendung wird, und übergibt die Verbindung an diesen Agenten. Bei B3 nimmt der Koordinatoragent Kontakt mit der fernen Clientanwendung auf und wird mit der Datenbank verbunden.

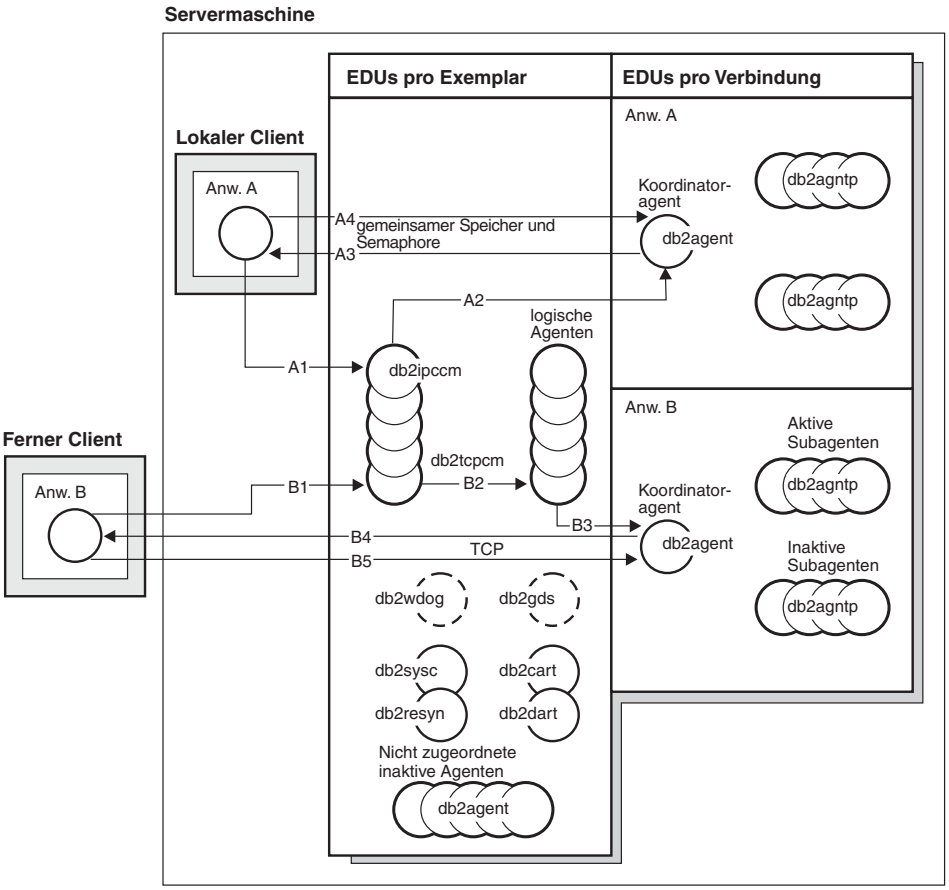


Abbildung 8. Übersicht über das Prozessmodell

Ferner sind folgende Details dieser Abbildung zu beachten:

- Arbeitsagenten führen Anwendungsanforderungen aus.
- Es gibt vier Typen von Arbeitsagenten: aktive Koordinatoragenten, aktive Subagenten, zugeordnete Subagenten und nicht zugeordnete Agenten in Bereitschaft.
- Jede Clientverbindung wird mit einem aktiven Koordinatoragenten verknüpft.
- In einer Umgebung mit partitionierten Datenbanken sowie in Umgebungen mit aktivierter partitionsinterner Parallelität verteilen die Koordinatoragenten Datenbankanforderungen an Subagenten (db2agntp). Die Subagenten führen die Anforderungen für die jeweilige Anwendung aus.
- Es gibt einen Agentenpool (db2agent), in dem inaktive Agenten zusammengefasst im Bereitschaftsmodus auf neue Arbeit warten.
- Andere EDUs verwalten Clientverbindungen, Protokolle, zweiphasige Festschreibungen (COMMIT), Sicherungs- und Wiederherstellungstasks sowie andere Tasks.

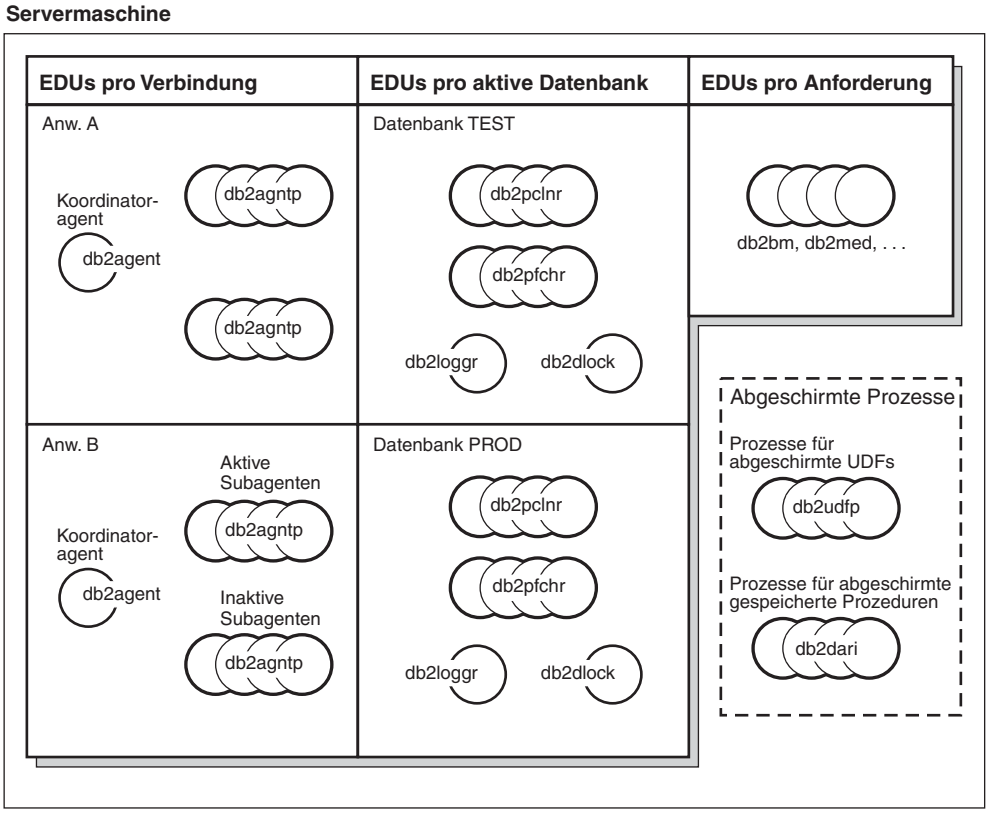


Abbildung 9. Prozessmodell, Teil 2

In dieser Abbildung werden zusätzliche EDUs (Engine Dispatchable Units, zuteilbare Einheiten der Steuerkomponente) gezeigt, die Teil der Servermaschinenumgebung sind. Alle aktiven Datenbanken verfügen über eigene gemeinsame Pools mit Vorablesefunktionen (db2pfchr) und Seitenlöschfunktionen (db2pclnr) sowie eigene Protokollfunktionen (db2loggr) und Detektoren für gegenseitiges Sperren (db2dlock).

Abgeschirmte benutzerdefinierte Funktionen (UDFs) und gespeicherte Prozeduren, die in der Abbildung nicht gezeigt werden, werden verwaltet, um den Aufwand zu minimieren, der mit ihrer Erstellung und Löschung verbunden ist. Der Standardwert für den Konfigurationsparameter *keepfenced* des Datenbankmanagers ist „YES", wodurch der Prozess der gespeicherten Prozedur zur erneuten Verwendung beim nächsten Aufruf einer gespeicherten Prozedur verfügbar bleibt.

Anmerkung: Nicht abgeschirmte benutzerdefinierte Funktionen (UDFs) und gespeicherte Prozeduren werden zwecks besserer Leistung direkt im Adressraum des Agenten ausgeführt. Da diese UDFs und gespeicherten Prozeduren jedoch über unbeschränkten Zugriff auf den Adressraum des Agenten verfügen, müssen sie vor ihrer Verwendung umfassend getestet werden.

Das Prozessmodell mit mehreren Partitionen ist eine logische Erweiterung des Prozessmodells mit einer einzigen Partition. Beide Betriebsarten werden durch eine gemeinsame Codebasis unterstützt. Die folgende Abbildung stellt die Ähnlichkeiten und Unterschiede zwischen dem Prozessmodell mit einer einzigen Partition, wie in den beiden vorangehenden Abbildungen gezeigt, und dem Prozessmodell mit mehreren Partitionen dar.

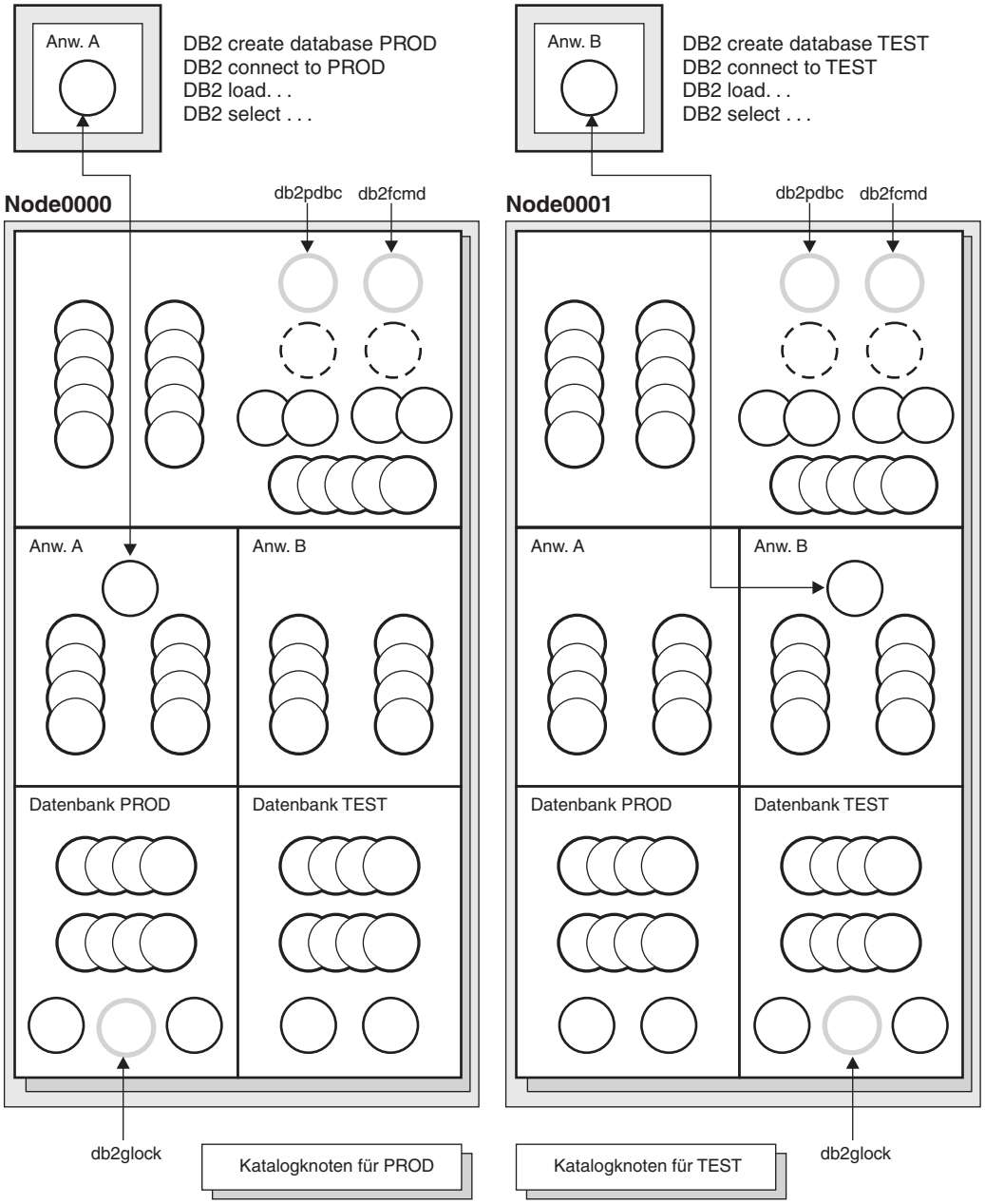


Abbildung 10. Prozessmodell und mehrere Partitionen

Die meisten EDUs stimmen zwischen dem Prozessmodell mit einer Partition und dem Prozessmodell mit mehreren Partitionen überein.

In einer Umgebung mit mehreren Partitionen (bzw. Knoten) ist eine der Partitionen der Katalogknoten. Der Katalog speichert alle Information in Bezug auf Objekte in der Datenbank.

Wie in der obigen Abbildung gezeigt, wird der Katalog der PROD-Datenbank im Knoten Node0000 erstellt, weil Anwendung A die PROD-Datenbank auf diesem Knoten erstellt. Analog wird der Katalog für die TEST-Datenbank im Knoten Node0001 erstellt, weil Anwendung B die TEST-Datenbank auf diesem Knoten erstellt.

Die Datenbanken sollten in verschiedenen Knoten erstellt werden, um so die zusätzlichen mit den Katalogen verbundenen Aktivitäten für jede Datenbank gleichmäßig auf die Knoten in der Systemumgebung zu verteilen.

Dem Exemplar sind zusätzliche EDUs (db2pdbc und db2fcmd) zugeordnet, die sich in jedem Knoten einer Datenbankumgebung mit mehreren Partitionen befinden. Diese EDUs werden für die Koordination von Anforderungen zwischen Datenbankpartitionen und zum Aktivieren des FCM (Fast Communications Manager) benötigt.

Dem Katalogknoten der Datenbank ist ebenfalls eine zusätzliche EDU (db2glock) zugeordnet. Diese EDU steuert globale Sperren für die Knoten, in denen sich die aktive Datenbank befindet.

Jede CONNECT-Verbindung von einer Anwendung wird durch eine Verbindung dargestellt, die einem Koordinatoragenten zur Verwaltung der Verbindung zugeordnet ist. Der *Koordinatoragent* ist der Agent, der mit der Anwendung kommuniziert, indem er Anforderungen empfängt und Antworten sendet. Er kann die Anforderung selbst erfüllen oder mehrere Subagenten zur Bearbeitung der Anforderung koordinieren. Die Partition, in der sich der Koordinatoragent befindet, wird als *Koordinatorknoten* dieser Anwendung bezeichnet. Der Koordinatorknoten kann auch mit dem Befehl SET CLIENT CONNECT_NODE festgelegt werden.

Teile der Datenbankanforderungen von der Anwendung werden vom Koordinatorknoten an Subagenten der anderen Partitionen gesendet. Alle Ergebnisse der anderen Partitionen werden auf dem Koordinatorknoten zusammengeführt, bevor sie zurück zur Anwendung gesendet werden.

Die Datenbankpartition, in der der Befehl CREATE DATABASE ausgeführt wurde, wird „Katalogknoten" der Datenbank genannt. In dieser Datenbankpartition werden die Katalogtabellen gespeichert. Normalerweise werden alle Benutzertabellen über eine Gruppe von Knoten partitioniert.

Anmerkung: Eine beliebige Anzahl von Partitionen kann zur Ausführung auf einer einzigen Maschine konfiguriert werden. Dies wird als Konfiguration mit „mehreren logischen Partitionen" oder mit „mehreren logischen Knoten" bezeichnet. Eine solche Konfiguration ist bei großen SMP-Maschinen (SMP - Symmetric Multiprocessor) mit einem sehr großen Hauptspeicher sehr nützlich. In einer solchen Umgebung kann die Kommunikation zwischen Partitionen durch die Verwendung von gemeinsam benutztem Speicher und Semaphors optimiert werden.

Zugehörige Konzepte:

- „Übersicht über die Architektur und Prozesse von DB2" auf Seite 9
- „Protokollverarbeitung" auf Seite 28
- „Aktualisierungsverarbeitung" auf Seite 31
- „Speicherverwaltung" auf Seite 38
- „Verbesserungen des Verbindungskonzentrators für Clientverbindungen" auf Seite 306

Speicherverwaltung

Eine Hauptaufgabe der Leistungsoptimierung besteht in der Entscheidung, wie der verfügbare Speicher unter den Bereichen innerhalb der Datenbank zu verteilen ist. Diese Verteilung des Speichers stimmen Sie in erster Linie durch die in diesem Abschnitt beschriebenen Konfigurationsparameter ab.

Alle EDUs (Engine Dispatchable Unit, zuteilbare Einheit der Steuerkomponente) in einer Partition sind mit dem gemeinsamen Speicher des Exemplars verbunden. Alle in einer Datenbank aktiven EDUs sind mit dem gemeinsamen Speicher dieser Datenbank verbunden. Alle für eine bestimmte Anwendung aktiven EDUs sind mit einer Region des gemeinsamen Speichers für diese Anwendung verbunden. Diese Art des gemeinsamen Speichers wird nur zugeordnet, wenn eine partitionsinterne oder partitionsübergreifende Parallelität aktiviert ist. Und schließlich verfügen alle EDUs über einen eigenen privaten Speicher.

Gemeinsam benutzter Speicher des Exemplars (auch als gemeinsam benutzter Speicher des Datenbankmanagers bezeichnet) wird beim Starten der Datenbank zugeordnet. Alle anderen Arten von Speicher werden aus dem gemeinsamen Speicher des Exemplars verbunden oder zugeordnet. Wenn der FCM (Fast Communications Manager) verwendet wird, werden diesem Speicher Puffer entnommen. Der FCM wird für die interne Kommunikation, in erster Linie Nachrichten, zwischen und innerhalb der Datenbankserver in einer bestimmten Datenbankumgebung verwendet. Wenn die erste Anwendung die Verbindung zu einer Datenbank herstellt, werden gemeinsam benutzte Speicherbereiche der Datenbank, gemeinsame Anwendungsspeicherbereiche und private Agentenspeicherbereiche zugeordnet. Der gemeinsame Exemplarspeicher kann durch den Konfigurationsparameter *instance_memory* gesteuert werden. Standardmäßig ist dieser Parameter auf `automatic` gesetzt, so dass DB2® die Größe des für das Exemplar zugeordneten Speichers berechnet.

Gemeinsam benutzter Speicher der Datenbank (auch als globaler Datenbankspeicher bezeichnet) wird zugeordnet, wenn eine Datenbank aktiviert oder zum ersten Mal eine Verbindung zu ihr hergestellt wird. Dieser Speicher wird für alle Anwendungen verwendet, die möglicherweise eine Verbindung zur Datenbank herstellen. Der gemeinsame Datenbankspeicher kann durch den Konfigurationsparameter *database_memory* gesteuert werden. Standardmäßig ist dieser Parameter auf *automatic* gesetzt, so dass DB2 die Größe des für die Datenbank zugeordneten Speichers berechnet.

In dem gemeinsamen Speicher einer Datenbank sind viele verschiedene Speicherbereiche enthalten. Dazu gehören:

- Pufferpools
- Sperrenliste
- Datenbankzwischenspeicher, wozu auch der Protokollpuffer zählt .
- Zwischenspeicher für Dienstprogramme
- Paketcache
- Katalogcache

Anmerkung: Speicher kann zugeordnet, freigegeben und zwischen verschiedenen Bereichen ausgetauscht werden, während die Datenbank aktiv ist. Zum Beispiel können Sie den Katalogcache verkleinern und anschließend einen bestimmten Pufferpool um den gleichen Betrag vergrößern. Bevor Sie die Konfigurationsparameter jedoch dynamisch ändern, müssen Sie eine Verbindung zu dieser Datenbank herstellen. Alle oben aufgeführten Speicherbereiche können dynamisch nur vergrößert, nicht verkleinert werden.

Der Konfigurationsparameter *numdb* des Datenbankmanagers gibt die Zahl der lokalen Datenbanken an, die gleichzeitig aktiv sein können. Der Wert des Parameters *numdb* kann Auswirkungen auf die Gesamtmenge an Speicher haben, die zugeordnet wird.

Gemeinsamer Anwendungsspeicher (auch als globaler Anwendungsspeicher bezeichnet) wird nur zugeordnet, wenn eine Anwendung die Verbindung zu einer Datenbank in einer partitionierten Datenbankumgebung, in einer nicht partitionierten Datenbankumgebung mit aktivierter partitionsinterner Parallelität oder bei aktiviertem Verbindungskonzentrator herstellt. Dieser Speicher wird von Agenten verwendet, die die Arbeit erledigen, die von mit der Datenbank verbundenen Clients angefordert wird.

Der Konfigurationsparameter *max_connections* des Datenbankmanagers legt eine obere Begrenzung für die Anzahl der Anwendungen fest, die eine Verbindung zu einer Datenbank herstellen können. Da jede Anwendung, die die Verbindung zu einer Datenbank herstellt, die Zuordnung von Speicher erforderlich macht, entsteht durch das Zulassen einer höheren Anzahl gleichzeitiger Anwendungen potenziell ein höherer Speicherbedarf.

Bis zu einem gewissen Grad wird die maximale Anzahl von Anwendungen auch durch den Konfigurationsparameter *maxagents* des Datenbankmanagers oder, bei Umgebungen mit partitionierten Datenbanken, durch den Parameter *max_coordagents* bestimmt. Der Parameter *maxagents* legt eine obere Begrenzung für die Gesamtzahl an Datenbankmanageragenten in einer Partition fest. Zu diesen Datenbankmanageragenten gehören aktive Koordinatoragenten, Subagenten, inaktive Agenten sowie nicht zugeordnete Agenten im Bereitschaftsmodus.

Ein privater Agentenspeicher wird für einen Agenten zugeordnet, wenn dieser Agent erstellt wird. Der private Agentenspeicher enthält Speicherzuordnungen, die nur für diesen bestimmten Agenten verwendet werden, wie zum Beispiel den Sortierspeicher und den Zwischenspeicher für Anwendungen.

Es gibt einige spezielle Arten des gemeinsamen Speichers:

- Gemeinsamer Speicher für Agent / lokale Anwendung. Dieser Speicher wird zur Kommunikation von SQL-Anforderungen und -Antworten zwischen einem Agenten und der zugehörigen Clientanwendung verwendet.
- Gemeinsamer Speicher für UDF/Agent. Zu diesem Speicher wird von Agenten eine Verbindung hergestellt, die eine abgeschirmte UDF oder gespeicherte Prozedur ausführen. Er wird als Kommunikationsbereich verwendet.
- Erweiterter Speicher. Ein normalerweise sehr großer (größer als 4 GB) Bereich von gemeinsam benutztem Speicher, der als erweiterter Pufferpool verwendet wird. Agenten, Vorablesefunktionen oder Seitenlöschfunktionen sind nicht permanent mit diesem Speicher verbunden, sondern stellen je nach Bedarf eine Verbindung zu einzelnen darin enthaltenen Segmenten her.

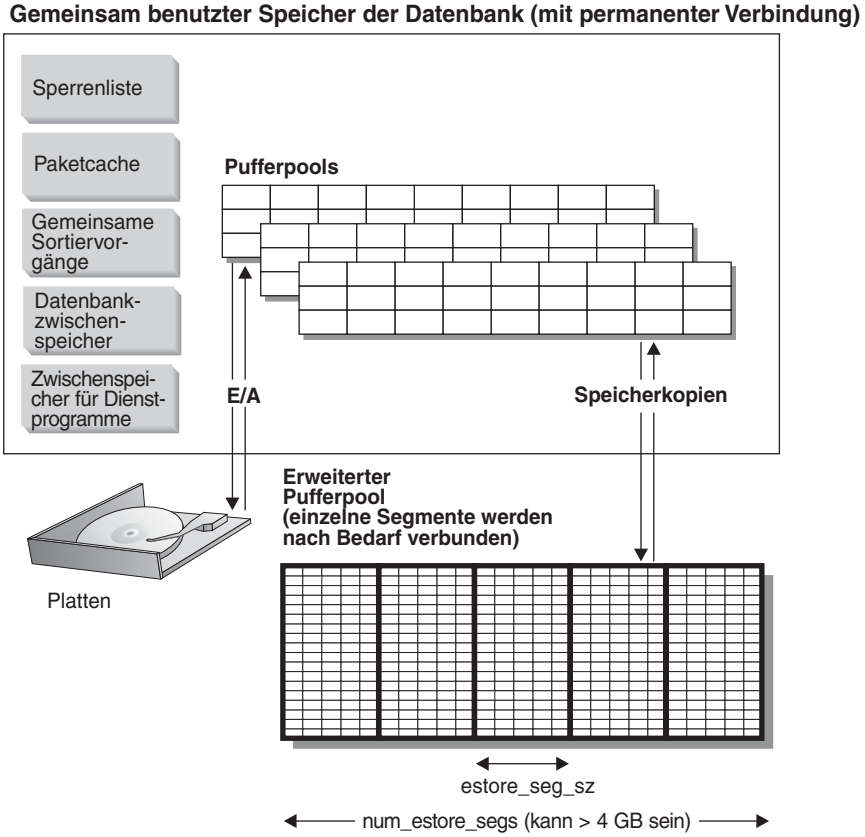


Abbildung 11. Verwendung des erweiterten Speichers durch Pufferpools

Der erweiterte Speicher dient als erweiterter Lookasidepuffer für die Hauptpufferpools. Er kann wesentlich größer als 4 GB sein. Für 32-Bit-Computer mit großen Hauptspeicherkapazitäten können Lookasidepuffer solche Verbesserungen der Speicherleistung ausnutzen. Der Erweiterungsspeichercache wird in Speichersegmenten definiert. Für 64-Bit-Computer sind solche Methoden für den Zugriff auf den gesamten verfügbaren Speicher nicht erforderlich.

Wenn Sie einen Teil des adressierbaren Realspeichers als Erweiterungsspeichercache verwenden, beachten Sie jedoch, dass dieser Speicher nicht mehr für andere Zwecke auf der Maschine genutzt werden kann, wie zum Beispiel als JFS-Cache (JFS - Journaled File System) oder als privater Adressraum für Prozesse. Das Zuordnen zusätzlichen adressierbaren Realspeichers zum Erweiterungsspeichercache führt möglicherweise zu höherem Aufkommen an System-Paging.

Die folgenden Datenbankkonfigurationsparameter beeinflussen die Anzahl und die Größe der für den Erweiterungsspeichercache verfügbaren Speichersegmente:

- Der Parameter *num_estore_segs* definiert die Zahl der Speichersegmente für den erweiterten Speicher.
- Der Parameter *estore_seg_sz* definiert die Größe der einzelnen Segmente für den erweiterten Speicher.

Jedem Tabellenbereich wird ein Pufferpool zugeordnet. Ein erweiterter Speichercache muss immer einem oder mehreren bestimmten Pufferpools zugeordnet werden. Die Seitengröße des erweiterten Speichercache muss mit der Seitengröße des Pufferpools übereinstimmen, dem er zugeordnet ist.

Zugehörige Konzepte:

- „Organisation der Speichernutzung" auf Seite 247
- „Gemeinsam benutzter Speicher des Datenbankmanagers" auf Seite 250
- „Globaler Speicher und zugehörige Steuerparameter" auf Seite 253
- „Pufferpoolverwaltung" auf Seite 257
- „Sekundäre Pufferpools im erweiterten Speicher auf 32-Bit-Plattformen" auf Seite 259
- „Richtlinien zur Optimierung von Parametern mit Auswirkung auf die Speichernutzung" auf Seite 255
- „Verbesserungen des Verbindungskonzentrators für Clientverbindungen" auf Seite 306

Zugehörige Referenzen:

- „estore_seg_sz - Segmentgröße für erweiterten Speicher" auf Seite 438
- „max_coordagents - Maximale Anzahl koordinierender Agenten" auf Seite 445
- „num_estore_segs - Segmentanzahl für erweiterten Speicher" auf Seite 438
- „maxagents - Maximale Anzahl von Agenten" auf Seite 446
- „numdb - Maximale Anzahl gleichzeitig aktiver Datenbanken einschließlich Host- und iSeries-Datenbanken" auf Seite 533
- „max_connections - Maximale Anzahl von Clientverbindungen" auf Seite 444
- „instance_memory - Exemplarspeicher" auf Seite 428
- „database_memory - Größe des gemeinsam benutzten Datenbankspeichers" auf Seite 398

Teil 2. Optimieren der Anwendungsleistung

Kapitel 3. Überlegungen zu Anwendungen

Eine Reihe von Faktoren kann die Leistung Ihrer Anwendung zur Laufzeit beeinflussen. Dieses Kapitel beschreibt einige der Faktoren, die beim Entwurf und der Codierung einer Anwendung und später bei der Optimierung dieser Anwendung zu beachten sind.

Steuerung des gemeinsamen Zugriffs und der Isolationsstufen

Die folgenden Abschnitte beschreiben die Auswirkung verschiedener Isolationsstufen auf den gemeinsamen Zugriff.

Aspekte des gemeinsamen Zugriffs

Da viele Benutzer auf Daten in einer relationalen Datenbank zugreifen und sie ändern, muss der Datenbankmanager in der Lage sein, einerseits den Benutzern diese Änderungen zu ermöglichen, und andererseits sicherzustellen, dass die Datenintegrität gewahrt bleibt. Der Begriff *gemeinsamer Zugriff* bezeichnet die Möglichkeit, dass mehrere interaktive Benutzer oder Anwendungsprogramme die Ressourcen zur gleichen Zeit verwenden können. Der Datenbankmanager steuert diesen Zugriff, um unerwünschte Folgen zu vermeiden. Solche Folgen sind zum Beispiel:

- **Verlorene Aktualisierungen**. Zwei Anwendungen A und B lesen beispielsweise dieselbe Zeile aus einer Datenbank und berechnen aufgrund der Daten, die von den Anwendungen gelesen werden, neue Werte für eine ihrer Spalten. Die Anwendung A aktualisiert die Zeile mit ihrem neuen Wert und anschließend aktualisiert B die Zeile ebenfalls, so dass die von A durchgeführte Aktualisierung verloren geht.
- **Zugriff auf nicht festgeschriebene Daten**. Anwendung A könnte zum Beispiel einen Wert in der Datenbank aktualisieren, den Anwendung B liest, bevor er festgeschrieben wurde. Wenn dann der Wert von der Anwendung A später nicht festgeschrieben, sondern rückgängig gemacht wird, basieren die Berechnungen der Anwendung B auf nicht festgeschriebenen (und daher wahrscheinlich ungültigen) Daten.
- **Nichtwiederholbare Lesevorgänge**. Einige Anwendungen bewirken die folgende Reihenfolge von Ereignissen: Anwendung A liest eine Zeile aus der Datenbank und verarbeitet anschließend zunächst andere SQL-Anforderungen. In der Zwischenzeit ändert oder löscht Anwendung B die Zeile und schreibt diese Aktion fest. Später, wenn Anwendung A versucht, die ursprüngliche Zeile erneut zu lesen, empfängt sie die geänderte Zeile oder stellt fest, dass die ursprüngliche Zeile gelöscht wurde.
- **Das Phänomen der Phantomzeilen**. Phantomzeilen treten unter folgenden Umständen auf:
  1. Eine Anwendung führt eine Abfrage aus, die eine Menge von Zeilen nach einer Suchbedingung liest.
  2. Eine andere Anwendung fügt neue Daten ein oder aktualisiert vorhandene Daten, die die Abfragebedingungen der ersten Anwendung erfüllen würden.
  3. Die erste Anwendung wiederholt die Abfrage aus Schritt 1 (innerhalb derselben Arbeitseinheit).

Einige zusätzliche Zeilen („Phantomzeilen") werden als Teil der Ergebnismenge zurückgegeben, die bei der ersten Ausführung der Abfrage (Schritt 1) nicht zurückgegeben wurden.

Anmerkung: Bei deklarierten temporären Tabellen gibt es keine Probleme bezüglich des gemeinsamen Zugriffs, da sie nur für die Anwendung verfügbar sind, die sie deklariert hat. Diese Art von Tabelle besteht nur von dem Zeitpunkt ihrer Deklaration durch die Anwendung bis zu dem Zeitpunkt, zu dem die Anwendung beendet wird oder die Verbindung trennt.

Steuerung des gemeinsamen Zugriffs in Systemen zusammengeschlossener Datenbanken

Ein *System zusammengeschlossener Datenbanken* unterstützt von Anwendungen und Benutzern übergebene SQL-Anweisungen, die auf zwei oder mehr Datenbankverwaltungssysteme (DBMS) bzw. Datenbanken innerhalb einer Anweisung verweisen. Als Verweise auf die Datenquellen, die aus einem DBMS und Daten bestehen, verwendet DB2® so genannte *Kurznamen*. Kurznamen sind Aliasnamen für Objekte in anderen Datenbankmanagern. In einem zusammengeschlossenen System verlässt sich DB2 auf die Protokolle zur Steuerung des gemeinsamen Zugriffs der Datenbankmanager, die die angeforderten Daten bereitstellen.

In einem DB2-System zusammengeschlossener Datenbanken besteht für Datenbankobjekte *Positionstransparenz*. Durch die Positionstransparenz ist es möglich, wenn Informationen von Tabellen und Sichten an andere Positionen versetzt werden, die Verweise auf diese Informationen über Kurznamen zu aktualisieren, ohne die Anwendungen ändern zu müssen, die diese Informationen anfordern. Wenn eine Anwendung auf Daten über Kurznamen zugreift, verlässt sich DB2 darauf, dass die Protokolle zur Steuerung des gemeinsamen Zugriffs der Datenbankmanager der Datenquellen die Isolationsstufen sicherstellen. Obwohl DB2 versucht, die angeforderte Isolationsstufe an der Datenquelle mit einem logischen Äquivalent abzugleichen, können die Ergebnisse je nach Funktionsspektrum der Datenquelle unterschiedlich sein.

Zugehörige Konzepte:

- „Auswirkungen von Isolationsstufen auf die Leistung" auf Seite 46

Zugehörige Tasks:

- „Angeben der Isolationsstufe" auf Seite 50

Zugehörige Referenzen:

- „locklist - Maximaler Speicher für Sperrenliste" auf Seite 399
- „maxlocks - Maximale Anzahl Sperren pro Anwendung" auf Seite 433

Auswirkungen von Isolationsstufen auf die Leistung

Eine *Isolationsstufe* legt fest, wie Daten gesperrt oder von anderen Prozessen während des Zugriffs auf die Daten isoliert werden. Die Isolationsstufe ist während der Dauer der Arbeitseinheit in Kraft. Anwendungen, die einen Cursor verwenden, der mit der Anweisung DECLARE CURSOR und mit der Klausel WITH HOLD deklariert wurde, behalten die gewählte Isolationsstufe für die Dauer der Arbeitseinheit bei, in der die Anweisung OPEN CURSOR durchgeführt wurde. DB2® unterstützt die folgenden Isolationsstufen:

- Wiederholtes Lesen (Repeatable Read)

- Lesestabilität (Read Stability)
- Cursorstabilität (Cursor Stability)
- Nicht festgeschriebenes Lesen (Uncommitted Read)

Anmerkung: Einige Hostdatenbankserver unterstützen die Isolationsstufe *kein Festschreiben* (No Commit). In anderen Datenbanken verhält sich diese Isolationsstufe wie die Isolationsstufe *Nicht festgeschriebenes Lesen* (Uncommited Read).

Im Folgenden sind detaillierte Erläuterungen zu den Isolationsstufen aufgeführt. Diese sind in absteigender Reihenfolge bezüglich der Auswirkungen auf die Leistung, aber in aufsteigender Reihenfolge bezüglich der Vorsicht geordnet, die beim Zugriff auf und Aktualisieren von Daten erforderlich ist.

Wiederholtes Lesen (Repeatable Read)

Die Isolationsstufe *Wiederholtes Lesen* (RR - Repeatable Read) sperrt alle Zeilen, auf die eine Anwendung innerhalb einer Arbeitseinheit verweist. Bei Verwendung der Isolationsstufe RR liefert eine Anweisung SELECT, die von einer Anwendung zweimal innerhalb derselben Arbeitseinheit, in der der Cursor geöffnet wurde, ausgegeben wird, jedes Mal dasselbe Ergebnis. Bei der Isolationsstufe RR können verlorene Aktualisierungen, Zugriffe auf nicht festgeschriebene Daten und Phantomzeilen nicht auftreten.

Eine Anwendung mit der Isolationsstufe RR kann, bis die Arbeitseinheit beendet wird, so oft wie erforderlich Zeilen abrufen und Aktionen an ihnen vornehmen. Andere Anwendungen können jedoch keine Zeile aktualisieren, löschen oder einfügen, die sich auf die Ergebnistabelle auswirken würde, bis die Arbeitseinheit beendet ist. Für Anwendungen mit der Isolationsstufe RR sind nicht festgeschriebene Änderungen anderer Anwendungen nicht sichtbar.

Durch die Isolationsstufe RR werden alle Zeilen, auf die verwiesen wird, und nicht nur die Zeilen, die abgerufen werden, gesperrt. Die Sperrung wird so ausgeführt, dass eine andere Anwendung keine Zeile einfügen oder aktualisieren kann, die der Liste von Zeilen, auf die in der Abfrage verwiesen wird, hinzugefügt würde, wenn die Abfrage erneut ausgeführt würde. Dadurch wird das Auftreten von Phantomzeilen verhindert. Wenn Sie zum Beispiel 10000 Zeilen unter Verwendung von Vergleichselementen durchsuchen, werden alle 10000 Zeilen gesperrt, auch wenn letzten Endes nur 10 Zeilen den Vergleichselementen entsprechen.

Anmerkung: Durch die Isolationsstufe RR wird sichergestellt, dass alle zurückgelieferten Daten bis zu dem Zeitpunkt, zu dem die Daten für die Anwendung *sichtbar* werden, unverändert bleiben, auch wenn temporäre Tabellen oder Zeilenblockung verwendet werden.

Da die Isolationsstufe RR eine beträchtliche Anzahl an Sperren erfordern kann, können diese Sperren die Anzahl von Sperren, die durch die Konfigurationsparameter *locklist* und *maxlocks* festgelegt wird, überschreiten. Um eine Sperreneskalation zu vermeiden, kann das Optimierungsprogramm eventuell von vornherein eine einzige Sperre auf Tabellenebene für eine Indexsuche anfordern, wenn das Auftreten einer Sperreneskalation wahrscheinlich ist. Dies funktioniert so, als würde der Datenbankmanager für Sie eine Anweisung LOCK TABLE absetzen. Wenn Sie keine Sperre auf Tabellenebene aktivieren lassen wollen, stellen Sie sicher, dass für die Transaktion ausreichend Sperren verfügbar sind, oder verwenden Sie die Isolationsstufe der Lesestabilität (RS).

Lesestabilität (RS)

Die Isolationsstufe *Lesestabilität* (RS - Read Stability) sperrt nur die Zeilen, die von einer Anwendung innerhalb einer Arbeitseinheit abgerufen werden. Sie stellt sicher, dass jede den Vergleichselementen entsprechende gelesene Zeile während einer Arbeitseinheit nicht von Prozessen anderer Anwendungen geändert wird und dass keine, von einem Prozess einer anderen Anwendung geänderte Zeile gelesen wird, bis die Änderung von dem Prozess festgeschrieben wurde. Das heißt, „nicht-wiederholbare Lesevorgänge" sind **nicht** möglich.

Anders als bei der Isolationsstufe RR (Wiederholtes Lesen) können bei der Isolationsstufe RS (Lesestabilität), wenn die Anwendung dieselbe Abfrage mehr als einmal absetzt, zusätzliche *Phantomzeilen* auftreten (*Phänomen der Phantomzeilen*). In dem bereits angeführten Beispiel der Suche über 10000 Zeilen werden durch die Isolationsstufe RS nur die den Vergleichselementen entsprechenden Zeilen gesperrt. Da durch die Abfrage nur 10 Zeilen abgerufen werden, werden unter der Isolationsstufe RS nur für diese zehn Zeilen Sperren aktiviert. Unter der Isolationsstufe RR (Wiederholtes Lesen) werden in diesem Beispiel hingegen sämtliche 10000 Zeilen gesperrt. Bei den aktivierten Sperren kann es sich um Sperren der Modi Share, Next Share, Update oder Exclusive handeln.

Anmerkung: Durch die Isolationsstufe RS wird sichergestellt, dass alle zurückgelieferten Daten bis zu dem Zeitpunkt, zu dem die Daten für die Anwendung *sichtbar* werden, unverändert bleiben, auch wenn temporäre Tabellen oder Zeilenblockung verwendet werden.

Ein Zweck der Isolationsstufe RS besteht darin, sowohl einen hohen Grad des gemeinsamen Zugriffs als auch eine stabile Sicht der Daten zur Verfügung zu stellen. Um diesen Zweck zu erfüllen, stellt das Optimierungsprogramm sicher, dass Sperren auf Tabellenebene erst aktiviert werden, wenn eine Sperreneskalation auftritt.

Die Isolationsstufe RS eignet sich am besten für Anwendungen, die alle folgenden Merkmale aufweisen:
- Sie arbeiten in einer Umgebung mit gemeinsamem Zugriff.
- Sie fordern Zeilen an, die für die Dauer der Arbeitseinheit stabil bleiben müssen.
- Sie verwenden dieselbe Abfrage in einer Arbeitseinheit nur einmal, oder es ist nicht notwendig, dass bei mehrmaliger Verwendung der Abfrage innerhalb derselben Arbeitseinheit stets dieselben Abfrageergebnisse erzielt werden.

Cursorstabilität (CS)

Die Isolationsstufe *Cursorstabilität* (CS - Cursor Stability) sperrt jede Zeile, auf die von einer Transaktion einer Anwendung zugegriffen wird, während sich der Cursor auf der Zeile befindet. Diese Sperre bleibt aktiv, bis die nächste Zeile abgerufen oder die Transaktion beendet wird. Wenn jedoch Daten in einer Zeile geändert werden, muss die Sperre aktiv bleiben, bis die Änderung in der Datenbank festgeschrieben wurde.

Keine anderen Anwendungen können eine Zeile aktualisieren oder löschen, die von einer Anwendung mit der Isolationsstufe CS abgerufen wurde, während sich ein Aktualisierungscursor auf der Zeile befindet. Für Anwendungen mit der Isolationsstufe CS sind nicht festgeschriebene Änderungen anderer Anwendungen nicht sichtbar.

In dem bereits angeführten Beispiel einer Suche über 10000 Zeilen heißt dies, dass unter der Isolationsstufe CS lediglich eine Sperre für die Zeile, die unter der aktuellen Cursorposition liegt, aktiv ist. Die Sperre wird entfernt, wenn der Cursor von der Zeile fortbewegt wird (sofern die Zeile nicht aktualisiert wurde).

Bei der Isolationsstufe CS können sowohl unterschiedliche Abfrageergebnisse innerhalb derselben Arbeitseinheit (d. h. nicht wiederholbare Lesevorgänge) als auch Phantomzeilen auftreten. Die Isolationsstufe CS ist die Standardisolationsstufe, die verwendet werden sollte, wenn es auf den höchstmöglichen Grad des gemeinsamen Zugriffs ankommt und nur festgeschriebene Zeilen anderer Anwendungen sichtbar sein sollen.

Nicht festgeschriebenes Lesen (UR)

Die Isolationsstufe *Nicht festgeschriebenes Lesen* (UR - Uncommitted Read) erlaubt einer Anwendung den Zugriff auf nicht festgeschriebene Änderungen anderer Transaktionen. Die Anwendung sperrt die Zeile, die sie liest, auch für keine andere Anwendung, sofern die andere Anwendung nicht versucht, die Tabelle zu löschen oder zu ändern. Die Isolationsstufe UR wirkt sich auf Lesecursor und Aktualisierungscursor unterschiedlich aus.

Cursor mit Lesezugriff können auf die meisten nicht festgeschriebenen Änderungen anderer Transaktionen zugreifen. Tabellen, Sichten und Indizes, die momentan von anderen Transaktionen erstellt oder gelöscht werden, sind während der Verarbeitung der Transaktion jedoch nicht verfügbar. Alle anderen Änderungen durch andere Transaktionen können gelesen werden, bevor sie festgeschrieben oder rückgängig gemacht wurden.

Anmerkung: Aktualisierungscursor, die unter der Isolationsstufe UR arbeiten, verhalten sich genauso wie unter der Isolationsstufe CS (Cursorstabilität).

Wenn eine Anwendung ein Programm unter der Isolationsstufe UR ausführt, kann sie die Isolationsstufe CS verwenden. Dies liegt an der Mehrdeutigkeit der Cursor, die in den Anwendungsprogrammen verwendet werden. Mehrdeutige Cursor können durch eine BLOCKING-Option auf die Isolationsstufe CS eskaliert werden. Der Standardwert der BLOCKING-Option ist UNAMBIG. Das bedeutet, dass mehrdeutige Cursor als aktualisierbar behandelt werden und die Eskalation der Isolationsstufe auf CS durchgeführt wird. Zur Vermeidung dieser Eskalation haben Sie die folgenden beiden Möglichkeiten:

- Ändern Sie die Cursor im Anwendungsprogramm, so dass die Mehrdeutigkeit beseitigt wird. Ändern Sie die SELECT-Anweisungen, um die Klausel FOR READ ONLY hinzuzufügen.
- Belassen Sie die mehrdeutigen Cursor im Anwendungsprogramm, aber kompilieren Sie das Programm mit der Option BLOCKING ALL vor oder binden Sie es mit dieser Option, um zuzulassen, dass alle mehrdeutigen Cursor als reine Lesecursor behandelt werden, wenn das Programm ausgeführt wird.

Wie in dem für die Isolationsstufe RR genannten Beispiel einer Suche über 10000 Zeilen werden bei der Isolationsstufe UR keine Zeilensperren aktiviert.

Bei der Isolationsstufe UR können sowohl unterschiedliche Leseergebnisse innerhalb derselben Arbeitseinheit (d. h. nicht wiederholbare Lesevorgänge) als auch Phantomzeilen auftreten. Die Isolationsstufe UR wird in der Regel für Abfragen auf Tabellen verwendet, die sich im Lesezugriff befinden, oder wenn Anweisungen

SELECT ausgeführt werden und es dabei keine Rolle spielt, ob nicht festgeschriebene Daten anderer Anwendungen angezeigt werden.

Zusammenfassung der Isolationsstufen

Die folgende Tabelle fasst die verschiedenen Isolationsstufen im Hinblick auf unerwünschte Nebeneffekte zusammen.

Tabelle 1. Zusammenfassung der Isolationsstufen

Isolationsstufe	Zugriff auf nicht festgeschriebene Daten	Nichtwiederholbare Lesevorgänge	Phänomen der Phantomzeilen
Wiederholtes Lesen (RR)	Nicht möglich	Nicht möglich	Nicht möglich
Lesestabilität (RS)	Nicht möglich	Nicht möglich	Möglich
Cursorstabilität (CS)	Nicht möglich	Möglich	Möglich
Nicht festgeschriebenes Lesen (UR)	Möglich	Möglich	Möglich

Die folgende Tabelle gibt einfache Anhaltspunkte, die Ihnen bei der Wahl der ersten Isolationsstufe für Ihre Anwendungen helfen können. Betrachten Sie die Angaben dieser Tabelle als Ausgangspunkt, und lesen Sie die Beschreibung der verschiedenen Isolationsstufen in den vorigen Abschnitten, um Hinweise zu erhalten, die vielleicht eine geeignetere Isolationsstufe empfehlen.

Tabelle 2. Richtlinien zur Wahl einer Isolationsstufe

Anwendungsart	Hohe Datenstabilität erforderlich	Hohe Datenstabilität nicht erforderlich
Schreib-/Lese-Transaktionen	RS	CS
Transaktionen mit Lesezugriff	RR oder RS	UR

Die Wahl der geeigneten Isolationsstufe für eine Anwendung ist zur Vermeidung von Erscheinungen, die für die Anwendung nicht tolerierbar sind, äußerst wichtig. Von der Isolationsstufe sind nicht nur die verschiedenen Grade der Isolation zwischen Anwendungen, sondern auch die Leistungsmerkmale einer einzelnen Anwendung betroffen, da die CPU- und Speicherressourcen, die zur Aktivierung und Freigabe von Sperren erforderlich sind, mit der Isolationsstufe variieren. Die Möglichkeit, dass gegenseitige Sperren auftreten, ist ebenfalls je nach Isolationsstufe unterschiedlich.

Zugehörige Konzepte:

- „Aspekte des gemeinsamen Zugriffs" auf Seite 45

Zugehörige Tasks:

- „Angeben der Isolationsstufe" auf Seite 50

Angeben der Isolationsstufe

Da die Isolationsstufe festlegt, wie Daten gesperrt und von anderen Prozessen während des Zugriffs auf die Daten isoliert werden, sollten Sie eine Isolationsstufe auswählen, die die Anforderungen des gemeinsamen Zugriffs und der Datenintegrität angemessen berücksichtigt. Die Isolationsstufe, die Sie angegeben, gilt für die Dauer der Arbeitseinheit.

Die Isolationsstufe kann auf verschiedene Arten angegeben werden. Die folgenden heuristischen Methoden können verwendet werden, um die Isolationsstufe festzulegen, die zum Kompilieren einer SQL-Anweisung verwendet wird:

Statisches SQL:

- Wenn eine Isolationsklausel in der Anweisung angegeben wird, wird der Wert dieser Klausel verwendet.
- Wird in der Anweisung keine Isolationsklausel angegeben, ist die verwendete Isolationsstufe die für das Paket zum Zeitpunkt der Bindung des Pakets an die Datenbank angegebene Klausel.

Dynamisches SQL:

- Wenn eine Isolationsklausel in der Anweisung angegeben wird, wird der Wert dieser Klausel verwendet.
- Wurde in der Anweisung keine Isolationsklausel angegeben, und wurde eine Anweisung SET CURRENT ISOLATION in der aktuellen Sitzung abgesetzt, wird der Wert des Sonderregisters CURRENT ISOLATION verwendet.
- Wird in der Anweisung keine Isolationsklausel angegeben, und wurde keine Anweisung SET CURRENT ISOLATION in der aktuellen Sitzung abgesetzt, ist die verwendete Isolationsstufe die für das Paket zum Zeitpunkt der Bindung des Pakets an die Datenbank angegebene Klausel.

Anmerkung: Viele kommerziell geschriebene Anwendungen ermöglichen eine Auswahl der Isolationsstufe. Informationen dazu finden Sie in der Dokumentation zur jeweiligen Anwendung.

Vorgehensweise:

Gehen Sie wie folgt vor, um die Isolationsstufe anzugeben:

1. **Beim Vorkompilieren oder Binden:**

   Für eine Anwendung, die in einer unterstützten kompilierten Sprache geschrieben ist, wird die Option ISOLATION der Befehle PREP oder BIND des Befehlszeilenprozessors verwendet. Sie können die Isolationsstufe auch über die APIs PREP und BIND angeben.

   - Wenn Sie beim Vorkompilieren eine Bindedatei erstellen, wird die Isolationsstufe in der Bindedatei gespeichert. Wenn Sie beim Binden keine Isolationsstufe angeben, wird standardmäßig die beim Vorkompilieren verwendete Isolationsstufe verwendet.
   - Wenn Sie keine Isolationsstufe angeben, wird die Standardisolationsstufe Cursorstabilität (CS) verwendet.

   Anmerkung: Führen Sie zur Feststellung der Isolationsstufe eines Pakets die folgende Abfrage aus:

   ```
   SELECT ISOLATION FROM SYSCAT.PACKAGES
     WHERE PKGNAME = 'XXXXXXXX'
     AND PKGSCHEMA = 'YYYYYYYY'
   ```

   Dabei steht *XXXXXXXX* für den Namen des Pakets und *YYYYYYYY* für den Namen des Schemas des Pakets. Beide Namen müssen vollständig in Großbuchstaben angegeben werden.

2. **Auf Datenbankservern, die REXX unterstützen:**

   Wenn eine Datenbank erstellt wird, werden mehrere Bindedateien, die die verschiedenen Isolationsstufen für SQL in REXX unterstützen, an die Datenbank

gebunden. Andere Befehlszeilenprozessorpakete werden ebenfalls an die Datenbank gebunden, wenn die Datenbank erstellt wird.

REXX und der Befehlszeilenprozessor stellen die Verbindung zu einer Datenbank mit der Standardisolationsstufe CS (Cursorstabilität) her. Durch die Änderung in eine andere Isolationsstufe wird der Status der Verbindung nicht geändert. Dies muss im Status CONNECTABLE AND UNCONNECTED oder IMPLICITLY CONNECTABLE ausgeführt werden.

Die Isolationsstufe, die von einer REXX-Anwendung verwendet wird, lässt sich durch Prüfen des Werts für die REXX-Variable SQLISL feststellen. Dieser Wert wird jedes Mal aktualisiert, wenn der Befehl CHANGE SQLISL ausgeführt wird.

3. **Auf Anweisungsebene:**

   Verwenden Sie die Klausel WITH. Die Isolationsstufe auf Anweisungsebene setzt die Isolationsstufe des Pakets außer Kraft, in dem die Anweisung auftritt.

   Für die folgenden SQL-Anweisungen können Sie eine Isolationsstufe angeben:

   - SELECT
   - SELECT INTO
   - DELETE mit Suche
   - INSERT
   - UPDATE mit Suche
   - DECLARE CURSOR

   Die folgenden Bedingungen gelten für Isolationsstufen, die für Anweisungen angegeben werden:

   - Die WITH-Klausel kann nicht in Unterabfragen verwendet werden.
   - Die Option WITH UR gilt nur für Operationen mit Lesezugriff. Anderenfalls wird die Anweisung automatisch von UR in CS geändert.

4. **Über CLI oder ODBC zur Laufzeit:**

   Verwenden Sie den Befehl CHANGE ISOLATION LEVEL. Für DB2 Call Level Interface (CLI) können Sie die Isolationsstufe bei der Konfiguration von CLI ändern. Verwenden Sie zur Laufzeit die Funktion *SQLSetConnectAttr* mit dem Attribut SQL_ATTR_TXN_ISOLATION, um die Transaktionsisolationsstufe für die aktuelle Verbindung zu definieren, auf die durch *ConnectionHandle* verwiesen wird.Darüber hinaus können Sie das Schlüsselwort TXNISOLATION in der Datei db2cli.ini verwenden.

5. **Über JDBC oder SQLJ zur Laufzeit:**

   Anmerkung: JDBC und SQLJ sind in DB2 mit CLI implementiert. Dies bedeutet, dass die Einstellungen der Datei ″db2cli.ini″ Einfluss auf die Projekte haben können, die unter Verwendung von JDBC und SQLJ geschrieben und ausgeführt werden.

   Verwenden Sie die Methode setTransactionIsolation() in der Schnittstellenverbindung java.sql.

   Erstellen Sie in SQLJ ein Paket mit Hilfe des SQLJ-Optimierungsprogramms db2profc. Zu den Optionen, die Sie für dieses Paket angeben können, gehört auch die Isolationsstufe.

6. **Für dynamisches SQL in der aktuellen Sitzung:**

   Verwenden Sie die Anweisung SET CURRENT ISOLATION, um die Isolationsstufe für dynamisches SQL festzulegen, die innerhalb einer Sitzung abgesetzt wird. Wird diese Anweisung abgesetzt, wird das Sonderregister CURRENT ISOLATION auf einen Wert festgelegt, der die Isolationsstufe für alle dynamischen SQL-Anweisungen angibt, die innerhalb der aktuellen Sitzung abgesetzt

wurden. Sobald dieser festgelegt wurde, stellt das Sonderregister CURRENT ISOLATION die Isolationsstufe für alle nachfolgenden dynamischen SQL-Anweisungen zur Verfügung, die innerhalb der Sitzung kompiliert wurden, unabhängig davon, welches Paket die Anweisung abgesetzt hat. Diese Isolationsstufe wird angewendet bis die Sitzung beendet wird oder bis eine Anweisung SET CURRENT ISOLATION mit der Option RESET abgesetzt wird.

Zugehörige Konzepte:

- „Aspekte des gemeinsamen Zugriffs" auf Seite 45

Zugehörige Referenzen:

- „SQLSetConnectAttr function (CLI) - Set connection attributes" in *CLI Guide and Reference, Volume 2*
- „CONNECT (Type 1) statement" in *SQL Reference, Volume 2*
- „Statement attributes (CLI) list" in *CLI Guide and Reference, Volume 2*

Steuerung des gemeinsamen Zugriffs und Sperren

Die folgenden Abschnitte beschreiben die verschiedenen Arten und Stufen der Sperrung und wie diese Sperren durch die Leistung des Datenzugriffs bestimmt und beeinflusst werden.

Sperren und Steuerung des gemeinsamen Zugriffs

Zur Steuerung des gemeinsamen Zugriffs sowie zur Verhinderung eines unkontrollierten Datenzugriffs richtet der Datenbankmanager Sperren für Pufferpools, Tabellen, Tabellenblöcke oder Tabellenzeilen ein. Eine *Sperre* ordnet eine Ressource des Datenbankmanagers einer Anwendung zu, die als *Sperreneigner* bezeichnet wird, um die Zugriffsmöglichkeiten anderer Anwendungen auf dieselbe Ressource zu steuern.

Die meisten Sperren werden für Tabellen erstellt. Beim Erstellen, Ändern oder Löschen von Pufferpools wird jedoch eine Pufferpoolsperre gesetzt. Mit dieser Sperre wird der Modus EXCLUSIVE (X) verwendet. Sie werden unter Umständen auf diese Sperre treffen, wenn eine Momentaufnahme über den Befehlszeilenprozessor (CLP) erstellt wird. Beim Anzeigen der Momentaufnahme wird ersichtlich, dass der Name der Sperre der Kennung (ID) des Pufferpools selbst entspricht.

Der Datenbankmanager verwendet Sperren entweder auf Datensatz- oder auf Tabellenebene, wobei folgende Faktoren die Grundlage für die Auswahl der jeweils geeigneten Sperre bilden:

- Die Isolationsstufe, die bei der Vorkompilierung oder beim Binden einer Anwendung an die Datenbank angegeben wird. Folgende Isolationsstufen sind möglich:
  - Nicht festgeschriebenes Lesen (UR)
  - Cursorstabilität (CS)
  - Lesestabilität (RS)
  - Wiederholtes Lesen (RR)

  Die unterschiedlichen Isolationsstufen werden zur Steuerung des Zugriffs auf nicht festgeschriebene Daten, zur Verhinderung verlorener Aktualisierungen, zur Ermöglichung nicht wiederholter Lesevorgänge für Daten sowie zur Verhinderung von Phantomzeilen verwendet. Verwenden Sie die minimale Isolationsstufe, die den Erfordernissen Ihrer jeweiligen Anwendung entspricht.

- Der vom Optimierungsprogramm ausgewählte Zugriffsplan. Tabellensuchen, Indexsuchen und andere Datenzugriffsmethoden erfordern jeweils verschiedene Arten des Zugriffs auf die Daten.
- Das Attribut LOCKSIZE für die Tabelle. Die Klausel LOCKSIZE in der Anweisung ALTER TABLE gibt die Unterteilung (Granularität) der Sperren an, die beim Zugriff auf die Tabellen verwendet werden. Ausgewählt werden können ROW für Zeilensperren oder TABLE für Tabellensperren. Verwenden Sie ALTER TABLE... LOCKSIZE TABLE für Tabellen, auf die lediglich Lesezugriff besteht. Dadurch wird die Zahl der Sperren verringert, die durch die Datenbankaktivität erforderlich werden.
- Der Speicherplatz, der für das Sperren aufgewendet wird. Der Speicherplatz, der für das Sperren aufgewendet wird, wird durch den Konfigurationsparameter `locklist` (Sperrenliste) der Datenbank gesteuert. Wenn sich die Sperrenliste füllt, kann sich die Leistung aufgrund von Sperreneskalationen und verringertem gemeinsamen Zugriff auf gemeinsam verwendete Objekte in der Datenbank verschlechtern. Wenn Sperreneskalationen häufig stattfinden, erhöhen Sie den Wert des Parameters `locklist`, des Parameters `maxlocks` oder die Werte beider Parameter.

Stellen Sie sicher, dass für alle Transaktionen häufig COMMIT-Operationen durchgeführt und so aktive Sperren freigegeben werden.

Im Allgemeinen werden Sperren auf Datensatzebene verwendet, sofern nicht einer der folgenden Fälle vorliegt:

- Die ausgewählte Isolationsstufe ist nicht festgeschriebenes Lesen (Uncommitted Read, UR).
- Die ausgewählte Isolationsstufe ist wiederholtes Lesen (Repeatable Read, RR) und der Zugriffsplan erfordert eine Suche ohne Vergleichselemente.
- Das Tabellenattribut ist „TABLE".
- Die Sperrenliste füllt sich und verursacht Sperreneskalation.
- Über die Anweisung LOCK TABLE erfolgt eine explizite Sperrung einer Tabelle. Die Anweisung LOCK TABLE verhindert, dass gleichzeitig ablaufende Anwendungsprozesse eine Tabelle ändern oder verwenden.

Eine *Sperreneskalation* findet statt, wenn die Anzahl von Sperren, die für Zeilen und Tabellen in der Datenbank aktiviert sind, dem Prozentsatz der Sperrenliste entspricht, der durch den Datenbankkonfigurationsparameter `maxlocks` definiert ist. Die Sperreneskalation kann für die Tabelle, in der die Sperre aktiviert wird, durch die die Eskalation ausgelöst wird, ohne Auswirkung bleiben. Zur Verringerung der Anzahl von Sperren auf ungefähr die Hälfte der Sperren, die beim Beginn der Sperreneskalation aktiviert sind, fängt der Datenbankmanager damit an, kleine Zeilensperren in Tabellensperren für alle aktiven Tabellen umzuwandeln, wobei er mit eventuell vorhandenen Sperren für LOB- oder LONG VARCHAR-Elemente beginnt. Eine *exklusive Sperreneskalation* ist eine Sperreneskalation, bei der die aktivierte Tabellensperre eine exklusive Sperre ist. Sperreneskalationen verringern den gemeinsamen Zugriff. Bedingungen, die Sperreneskalationen verursachen könnten, sollten vermieden werden.

Die Dauer einer Zeilensperre ist je nach verwendeter Isolationsstufe unterschiedlich:

- UR-Suchen: Zeilensperren werden nicht aktiviert, sofern keine Zeilen Daten geändert werden.
- CS-Suchen: Zeilensperren bleiben nur für die Zeit aktiviert, die sich der Cursor auf der Zeile befindet.

- RS-Suchen: Sperren bleiben nur für die Dauer der Transaktion für Zeilen aktiviert, die den Suchbedingungen entsprechen.
- RR-Suchen: Alle Zeilensperren bleiben für die Dauer der Transaktion aktiviert.

Zugehörige Konzepte:

- „Sperrenattribute" auf Seite 55
- „Sperren und Leistung" auf Seite 57
- „Richtlinien für Sperren" auf Seite 62

Zugehörige Referenzen:

- „locklist - Maximaler Speicher für Sperrenliste" auf Seite 399
- „dlchktime - Zeitintervall für Prüfung gegenseitiger Sperren" auf Seite 431
- „diaglevel - Aufzeichnungsebene bei Fehlerdiagnose" auf Seite 524
- „locktimeout - Zeitlimit für Sperren" auf Seite 432
- „Sperrentypenkompatibilität" auf Seite 69
- „Sperrmodi und Zugriffspfade für Standardtabellen" auf Seite 70
- „Sperrmodi für Tabellen- und Satz-ID-Indexsuchen für MDC-Tabellen" auf Seite 73
- „Sperren für Blockindexsuchen für MDC-Tabellen" auf Seite 77

Sperrenattribute

Sperren des Datenbankmanagers verfügen über folgende Grundattribute:

Modus
Die Art des Zugriffs, die dem Sperreneigner gewährt wird, sowie die Art des Zugriffs, die Benutzern gewährt wird, die das gesperrte Objekt gleichzeitig verwenden. Dies wird manchmal auch als *Status* der Sperre bezeichnet.

Objekt
Die Ressource, die gesperrt ist. Der einzige Typ von Objekt, den Sie explizit sperren können, ist eine Tabelle. Der Datenbankmanager implementiert Sperren auch für andere Arten von Ressourcen, wie Zeilen, Tabellen und Tabellenbereiche. Für MDC-Tabellen (Tabellen mit mehrdimensionalem Clustering) können außerdem Blocksperren aktiviert werden. Das Objekt, das gesperrt wird, bestimmt die *Granularität* der Sperre.

Dauer Der Zeitraum, für den eine Sperre aktiv ist. Die Isolationsstufe, in der die Abfrage ausgeführt wird, beeinflusst die Sperrdauer.

Die folgende Tabelle zeigt die Modi und ihre Auswirkungen in der Reihenfolge zunehmender Ressourcenbeschränkung. Detaillierte Informationen über Sperren auf verschiedenen Stufen finden Sie in den Referenztabellen zu den Sperrmodi.

Tabelle 3. Zusammenfassung der Sperrmodi

Sperrmodus	Gültiger Objekttyp	Beschreibung
IN (Intent None)	Tabellenbereiche, Blöcke, Tabellen	Der Sperreneigner kann alle Daten im Objekt lesen, einschließlich der nicht festgeschriebenen Daten, aber er kann keine Daten aktualisieren. Andere gleichzeitig ausgeführte Anwendungen können die Tabelle lesen oder aktualisieren.
IS (Intent Share)	Tabellenbereiche, Blöcke, Tabellen	Der Sperreneigner kann Daten in der gesperrten Tabelle lesen, aber nicht aktualisieren. Andere Anwendungen können die Tabelle lesen oder aktualisieren.

Tabelle 3. Zusammenfassung der Sperrmodi (Forts.)

Sperrmodus	Gültiger Objekttyp	Beschreibung
NS (Next Key Share)	Zeilen	Der Sperreneigner und jede gleichzeitig ausgeführte Anwendung kann die gesperrten Zeilen lesen, aber nicht aktualisieren. Diese Sperre wird anstatt einer Sperre im Modus S für Zeilen einer Tabelle aktiviert, wenn die Isolationsstufe der Anwendung entweder RS oder CS ist. Der Sperrmodus NS wird nicht zum Sperren der nächsten Zeile (Next-Key locking) verwendet. Er wird anstelle des Modus S bei CS- und RS-Suchen verwendet, um die Auswirkungen des Sperrens der nächsten Zeilen bei diesen Suchen zu minimieren.
S (Share)	Zeilen, Blöcke, Tabellen	Der Sperreneigner und jede gleichzeitig ausgeführte Anwendung kann die gesperrten Daten lesen, aber nicht aktualisieren.
IX (Intent Exclusive)	Tabellenbereiche, Blöcke, Tabellen	Der Sperreneigner und gleichzeitig ausgeführte Anwendungen können Daten lesen und aktualisieren. Andere gleichzeitig ausgeführte Anwendungen können die Tabelle lesen und auch aktualisieren.
SIX (Share with Intent Exclusive)	Tabellen, Blöcke	Der Sperreneigner kann Daten lesen und aktualisieren. Andere gleichzeitig ausgeführte Anwendungen können die Tabelle lesen.
U (Update)	Zeilen, Blöcke, Tabellen	Der Sperreneigner kann Daten aktualisieren. Andere Arbeitseinheiten können die Daten im gesperrten Objekt lesen, aber nicht versuchen, sie zu aktualisieren.
NW (Next Key Weak Exclusive)	Zeilen	Wenn eine Zeile in einen Index eingefügt wird, wird eine NW-Sperre für die nächste Zeile aktiviert. Für Indizes des Typs 2 findet dies nur statt, wenn die nächste Zeile momentan von einer RR-Suche gesperrt wird. Der Sperreneigner kann die gesperrte Zeile lesen, aber nicht aktualisieren. Dieser Sperrmodus ist dem Sperrmodus X ähnlich, ist jedoch auch mit den Sperrmodi W und NS kompatibel.
X (Exclusive)	Zeilen, Blöcke, Tabellen, Pufferpools	Der Sperreneigner kann Daten im gesperrten Objekt lesen und auch aktualisieren. Nur Anwendungen mit der Isolationsstufe UR (nicht festgeschriebenes Lesen) können auf das gesperrte Objekt zugreifen.
W (Weak Exclusive)	Zeilen	Eine Sperre in diesem Modus wird für die Zeile aktiviert, wenn eine Zeile in eine Tabelle eingefügt wird, für die keine Indizes des Typs 2 definiert sind. Der Sperreneigner kann die gesperrte Zeile ändern. Dieser Sperrmodus wird auch während des Einfügens in einen eindeutigen Index verwendet, um festzustellen, ob ein doppelter Wert festgeschrieben wurde, wenn ein doppelter Wert angetroffen wird. Dieser Sperrmodus ist dem Sperrmodus X ähnlich, jedoch ist er mit dem Sperrmodus NW kompatibel. Nur Anwendungen mit der Isolationsstufe UR (nicht festgeschriebenes Lesen) können auf die gesperrte Zeile zugreifen.
Z (Super Exclusive)	Tabellenbereiche, Tabellen	Diese Sperre wird für eine Tabelle unter bestimmten Umständen aktiviert, zum Beispiel, wenn die Tabelle geändert (ALTER) oder gelöscht (DROP) wird, ein Index für die Tabelle erstellt oder gelöscht wird oder bestimmte Arten von Tabellenreorganisation durchgeführt werden. Keine anderen gleichzeitig ausgeführten Anwendungen können die Tabelle lesen oder aktualisieren.

Zugehörige Konzepte:
- „Sperren und Steuerung des gemeinsamen Zugriffs" auf Seite 53
- „Sperren und Leistung" auf Seite 57

Zugehörige Referenzen:
- „maxlocks - Maximale Anzahl Sperren pro Anwendung" auf Seite 433
- „Sperrentypenkompatibilität" auf Seite 69
- „Sperrmodi und Zugriffspfade für Standardtabellen" auf Seite 70

Sperren und Leistung

Verschiedene zusammenhängende Faktoren beeinflussen die Verwendung von Sperren und deren Auswirkung auf die Anwendungsleistung. In diesem Abschnitt werden die folgenden Faktoren behandelt:

- Gemeinsamer Zugriff und Granularität
- Sperrenkompatibilität
- Sperrenumwandlung
- Sperreneskalation
- Wartezeiten und Zeitlimits für Sperren
- Gegenseitige Sperren

Gemeinsamer Zugriff und Granularität

Wenn eine Anwendung eine Sperre für ein Datenbankobjekt aktiviert hat, kann eine andere Anwendung auf dieses Objekt vielleicht nicht zugreifen. Aus diesem Grund sind Sperren auf Zeilenebene in Bezug auf einen maximalen gemeinsamen Zugriff besser als Sperren auf Tabellenebene. Jedoch benötigen Sperren Speicherplatz und Verarbeitungszeit, so dass eine einzelne Tabellensperre den Sperrenaufwand minimiert.

Die Klausel LOCKSIZE der Anweisung ALTER TABLE legt den Geltungsbereich (Granularität) von Sperren auf die Zeilen- oder Tabellenebene fest. Standardmäßig werden Zeilensperren verwendet. Nur Sperren der Modi S (Shared) und X (Exclusive) werden durch diese definierten Tabellensperren angefordert. Die Klausel LOCKSIZE ROW in der Anweisung ALTER TABLE verhindert normale Sperreneskalationen nicht.

Eine permanente Tabellensperre, die durch eine Anweisung ALTER TABLE definiert wird, kann einer Tabellensperre für eine einzelne Transaktion durch die Anweisung LOCK TABLE in folgenden Fällen vorzuziehen sein:

- Die Tabelle ist schreibgeschützt und benötigt immer nur S-Sperren. Andere Benutzer können ebenfalls S-Sperren für die Tabelle aktivieren.
- Auf die Tabelle greifen in der Regel reine Leseanwendungen zu, jedoch greift manchmal ein einzelner Benutzer zu kurzen Pflegemaßnahmen auf sie zu, und dieser Benutzer benötigt eine X-Sperre. Während das Pflegeprogramm ausgeführt wird, werden die reinen Leseanwendungen ausgesperrt. Unter anderen Umständen können reine Leseanwendung jedoch auf die Tabelle bei minimalem Sperrenaufwand gleichzeitig zugreifen.

Die Anweisung ALTER TABLE gibt Sperren global an, wodurch alle Anwendungen und Benutzer, die auf die Tabelle zugreifen, betroffen werden. Einzelne Anwendungen können die Anweisung LOCK TABLE verwenden, um stattdessen Tabellensperren auf Anwendungsebene zu definieren.

Sperrenkompatibilität

Die Sperrenkompatibilität ist ein weiterer wichtiger Faktor beim gleichzeitigen Zugriff auf Tabellen. Die Sperrenkompatibilität bezieht sich auf die aktuelle Sperre für ein Objekt und den Typ von Sperre, der für dieses Objekt angefordert wird, und bestimmt, ob diese Anforderung erfüllt werden kann.

Nehmen Sie an, dass Anwendung A eine Sperre für eine Tabelle aktiviert hat, auf die Anwendung B ebenfalls zugreifen will. Der Datenbankmanager fordert für die Anwendung B eine Sperre eines bestimmten Modus an. Wenn der Modus der für A aktiven Sperre die für B angeforderte Sperre zulässt, werden diese beiden Sperren (bzw. Modi) als kompatibel bezeichnet.

Wenn der für Anwendung B angeforderte Sperrmodus nicht mit der für Anwendung A aktiven Sperre kompatibel ist, kann Anwendung B nicht fortfahren. Stattdessen muss Anwendung B warten, bis Anwendung A die Sperre freigegeben hat und bis alle weiteren bestehenden inkompatiblen Sperren freigegeben wurden.

Sperrenumwandlung

Die Änderung des Modus einer bereits aktivierten Sperre wird als *Umwandlung* bezeichnet. Die Sperrenumwandlung erfolgt, wenn ein Prozess auf ein Datenobjekt zugreift, für das er bereits eine Sperre aktiviert hat, und der Zugriffsmodus eine noch stärker einschränkende als die aktuelle Sperre erfordert. Für einen Prozess kann immer nur eine Sperre für ein Datenobjekt aktiv sein, obwohl er mehrfach eine Sperre für dasselbe Datenobjekt indirekt durch eine Abfrage anfordern kann.

Einige Sperrmodi gelten nur für Tabellen, andere nur für Zeilen oder Blöcke. Für Zeilen oder Blöcke findet in der Regel eine Umwandlung statt, wenn eine X-Sperre benötigt wird und eine S- oder U-Sperre (Update) zurzeit aktiviert ist.

Sperren der Modi IX (Intent Exclusive) und S (Shared) stellen jedoch in Bezug auf die Sperrenumwandlung einen Sonderfall dar. Weder der Modus S noch der Modus IX wird als stärker einschränkend angesehen als der jeweils andere Modus. Wenn einer dieser Modi aktiv ist und der andere angefordert wird, erfolgt eine Sperrenumwandlung in den Modus SIX (Share with Intent Exclusive). Alle anderen Umwandlungen werden so ausgeführt, dass der angeforderte Sperrmodus zum Modus der aktiven Sperre wird, wenn der angeforderte Modus einen höheren Grad der Einschränkung bewirkt.

Es kann auch eine doppelte Umwandlung stattfinden, wenn eine Abfrage eine Zeile aktualisiert. Wenn die Zeile über einen Indexzugriff gelesen wird und im Modus S gesperrt ist, hat die Tabelle, die diese Zeile enthält, eine abdeckende Intent-Sperre. Wenn der Sperrentyp jedoch IS und nicht IX ist und die Zeile nachfolgend geändert wird, wird die Tabellensperre in eine IX-Sperre und die Zeilensperre in eine X-Sperre umgewandelt.

Eine Sperrenumwandlung findet normalerweise implizit bei der Ausführung einer Abfrage statt. Das Verständnis der verschiedenen Arten von Sperren für verschiedene Abfragen sowie Tabellen- und Indexkombinationen kann beim Entwurf und bei der Optimierung von Anwendungen hilfreich sein.

Sperreneskalation

Die Sperreneskalation ist ein interner Mechanismus, der die Anzahl der aktivierten Sperren verringert. In einer Tabelle werden viele Zeilensperren, bzw. bei MDC-Tabellen (Tabellen mir mehrdimensionalem Clustering) viele Zeilen- oder Blocksperren, in eine Tabellensperre hochgestuft. Die Sperreneskalation findet statt, wenn Anwendungen zu viele Sperren beliebiger Art aktiviert haben. Die Sperreneskalation kann für einen bestimmten Datenbankagenten auftreten, wenn der Agent seine Zuordnung der Sperrenliste überschreitet. Die Sperreneskalation wird intern durchgeführt. Das einzige extern bemerkbare Ergebnis könnte sein, dass der gemeinsame Zugriff auf eine oder mehrere Tabellen verringert wird.

In einer adäquat konfigurierten Datenbank tritt die Sperreneskalation nur selten auf. Eine Sperreneskalation kann beispielsweise auftreten, wenn ein Anwendungsprogrammierer einen Index für eine sehr große Tabelle erstellt, um die Leistung und den gemeinsamen Zugriff zu verbessern, eine Transaktion jedoch auf die meisten Datensätze in der Tabelle zugreift. Da der Datenbankmanager in diesem Fall nicht vorausberechnen kann, in welchem Umfang die Tabelle gesperrt wird, sperrt er jeden Datensatz einzeln, anstatt nur die Tabelle im Modus S oder X zu sperren. In diesem Fall kann sich der Datenbankdesigner mit dem Anwendungsentwickler besprechen und eine Anweisung LOCK TABLE für diese Transaktion empfehlen.

Gelegentlich kann es vorkommen, dass der Prozess, der die interne Eskalationsanforderung empfängt, nur wenige oder keine Zeilensperren für eine Tabelle aktiviert hat, die Sperren jedoch eskaliert werden, weil ein oder mehrere Prozesse zahlreiche Sperren aktiviert haben. Der Prozess fordert eventuell gar keine weitere Sperre an oder greift auf die Datenbank nur zu, um die Transaktion zu beenden. In diesem Fall fordert vielleicht ein anderer Prozess die Sperre oder die Sperren an, die die Eskalationsanforderung auslösen.

Anmerkung: Eine Sperreneskalation kann auch zu gegenseitigen Sperren führen. Nehmen Sie zum Beispiel an, eine reine Leseanwendung und eine Aktualisierungsanwendung greifen auf die gleiche Tabelle zu. Wenn die Aktualisierungsanwendung exklusive Sperren für viele Zeilen der Tabelle aktiviert hat, könnte der Datenbankmanager versuchen, die Sperren dieser Tabelle zu einer exklusiven Tabellensperre zu eskalieren. Jedoch bewirkt die Tabellensperre, die von der reinen Leseanwendung aktiviert wurde, dass die Anforderung zur Eskalation in eine exklusive Tabellensperre warten muss. Falls nun die reine Leseanwendung eine Zeilensperre für eine Zeile anfordert, die bereits von der Aktualisierungsanwendung gesperrt ist, entsteht eine gegenseitige Sperre. Zur Vermeidung dieser Art von Problem können Sie entweder die Aktualisierungsanwendung so codieren, dass sie die Tabelle beim Starten der Anwendung exklusiv sperrt, oder Sie können die Sperrenliste vergrößern.

Wartezeiten und Zeitlimits für Sperren

Die Erkennung von Zeitlimitüberschreitungen für Sperren ist eine Datenbankmanagerfunktion, die verhindert, dass Anwendungen unendlich lange auf die Freigabe einer Sperre in einer abnormalen Situation warten. Zum Beispiel könnte eine Transaktion auf eine Sperre warten, die von der Anwendung eines anderen Benutzers aktiviert wurde. Der andere Benutzer hat jedoch seine Workstation verlassen, ohne seine Anwendung die Transaktion festschreiben zu lassen, wodurch die Sperre freigegeben würde. Um die Blockierung einer Anwendung in einem solchen Fall zu vermeiden, setzen Sie den Konfigurationsparameter *locktimeout* auf die maximale Zeitdauer, die eine beliebige Anwendung auf eine Sperre warten sollte.

Durch die Einstellung dieses Parameters können globale gegenseitige Sperren besser vermieden werden, insbesondere in Anwendungen mit verteilten Arbeitseinheiten (DUOW). Wenn die Zeit, die eine Sperrenanforderung ansteht, länger ist als die im Wert des Parameters *locktimeout* definierte Zeit, empfängt die anfordernde Anwendung einen Fehler und ihre Transaktion wird rückgängig gemacht. Wenn zum Beispiel *Programm 1* versucht, eine Sperre zu erhalten, die bereits für *Programm 2* aktiv ist, liefert *Programm 1* den SQLCODE -911 (SQLSTATE 40001) mit dem Ursachencode 68 zurück, wenn das Zeitlimit abläuft. Der Standardwert für den Parameter *locktimeout* ist -1, d. h. die Erkennung der Überschreitung des Sperrzeitlimits ist ausgeschaltet.

Anmerkung: Für Tabellen, Zeilen- und MDC-Blocksperren kann eine Anwendung die Einstellung für *locktimeout* auf der Datenbankebene mit der Anweisung SET CURRENT LOCK TIMEOUT überschreiben.

Wenn Sie mehr Informationen über Zeitlimits für Sperrenanforderungen in den Benachrichtigungsprotokollen für die Systemverwaltung protokollieren wollen, setzen Sie den Konfigurationsparameter *notifylevel* des Datenbankmanagers auf den Wert 4. Die protokollierten Informationen geben das gesperrte Objekt, den Sperrmodus und die Anwendung an, die die Sperre aktiviert hat. Der Name der aktuellen dynamischen SQL-Anweisung oder des statischen Pakets kann ebenfalls protokolliert werden. Eine dynamische SQL-Anweisung wird nur bei Benachrichtigungsstufe (*notifylevel*) 4 protokolliert.

Gegenseitige Sperren

Konkurrenzsituationen bei der Anforderung von Sperren können zu gegenseitigen Sperren führen. Nehmen Sie zum Beispiel an, dass Prozess 1 die Tabelle A im Modus X (exklusiv) sperrt und Prozess 2 die Tabelle B im Modus X sperrt. Wenn Prozess 1 nun versucht, Tabelle B im Modus X zu sperren, und Prozess 2 versucht, Tabelle A im Modus X zu sperren, sperren sich die Prozesse gegenseitig. Bei einer gegenseitigen Sperre sind beide Prozesse so lange blockiert, bis die jeweils zweite Sperrenanforderung erfüllt wird, jedoch wird keine der beiden Anforderungen ausgeführt, sofern nicht einer der Prozesse eine COMMIT- oder ROLLBACK-Operation ausführt. Dieser Zustand dauert an, bis ein externer Agent einen der Prozesse aktiviert und dazu zwingt, eine ROLLBACK-Operation auszuführen.

Zur Behandlung gegenseitiger Sperren verwendet der Datenbankmanager den so genannten Detektor für gegenseitiges Sperren, der als asynchroner Systemprozess im Hintergrund ausgeführt wird. Der Detektor für gegenseitiges Sperren wird regelmäßig in den vom Konfigurationsparameter *dlchktime* definierten Intervallen aktiv. Wenn der Detektor für gegenseitiges Sperren aktiv wird, untersucht er das Sperrensystem auf gegenseitige Sperren. In einer partitionierten Datenbank sendet jede Partition *Sperrendiagramme* an die Datenbankpartition, die die Systemkatalogsichten enthält. Die Erkennung globaler gegenseitiger Sperren findet in dieser Partition statt.

Wenn eine gegenseitige Sperre festgestellt wird, wählt der Detektor einen der Prozesse als den *betroffenen Prozess* aus, der rückgängig zu machen ist. Der betroffene Prozess wird aktiviert und gibt den SQLCODE -911 (SQLSTATE 40001) mit Ursachencode 2 an die aufrufende Anwendung zurück. Der Datenbankmanager macht den ausgewählten Prozess automatisch rückgängig (ROLLBACK). Wenn die ROLLBACK-Operation beendet ist, werden die Sperren freigegeben, die zu dem betroffenen Prozess gehörten, und die anderen Prozesse, die ebenfalls an der gegenseitigen Sperre beteiligt waren, können fortfahren.

Zur Gewährleistung einer guten Leistung müssen Sie ein geeignetes Intervall für den Detektor für gegenseitiges Sperren auswählen. Ein zu kurzes Intervall verursacht einen unnötigen Systemaufwand, ein zu langes Intervall verlängert die Verzögerung eines Prozesses durch eine gegenseitige Sperre auf ein nicht tolerierbares Maß. Zum Beispiel lässt ein Aktivierungsintervall von fünf Minuten zu, dass eine gegenseitige Sperre annähernd fünf Minuten bestehen kann, was für eine kurze Transaktionsverarbeitung bereits ein langer Zeitraum sein kann. Wägen Sie die möglichen Verzögerungen durch die Auflösung gegenseitiger Sperren gegen den Aufwand zur ihrer Erkennung ab.

In einer partitionierten Datenbank wird das durch den Konfigurationsparameter *dlchktime* definierte Intervall nur auf dem Katalogknoten angewendet. Werden in einer partitionierten Datenbank zahlreiche gegenseitige Sperren festgestellt, erhöhen Sie den Wert des Parameters *dlchktime*, um den Wartezeiten für Sperren und Kommunikation Rechnung zu tragen.

Ein anderes Problem tritt auf, wenn eine Anwendung mit mehr als einem unabhängigen Prozess, der auf die Datenbank zugreift, so strukturiert ist, dass die Entstehung gegenseitiger Sperren wahrscheinlich ist. Ein Beispiel wäre eine Anwendung, in der mehrere Prozesse auf dieselbe Tabelle zuerst zu Leseoperationen und anschließend zu Schreiboperationen zugreifen. Wenn die Prozesse zuerst SQL-Abfragen im Lesezugriff durchführen und anschließend SQL-Aktualisierungsanweisungen für dieselbe Tabelle verarbeiten, steigt die Wahrscheinlichkeit gegenseitiger Sperren, weil es zwischen den Prozessen zu Konkurrenzsituationen beim Zugriff auf dieselben Daten kommen kann. Wenn zum Beispiel zwei Prozesse die Tabelle lesen und anschließend aktualisieren, könnte Prozess A versuchen, eine Sperre des Modus X für eine Zeile zu erhalten, für die Prozess B eine Sperre des Modus S hat, und umgekehrt. Zur Vermeidung solcher gegenseitigen Sperren, sollten Anwendungen, die auf Daten zugreifen, um diese zu ändern, auf eine der folgenden Arten verfahren:

- Verwenden der Klausel FOR UPDATE OF bei der Ausführung einer SELECT-Anweisung. Durch diese Klausel wird sichergestellt, dass eine Sperre des Modus U aktiviert wird, wenn der Prozess A versucht, die Daten zu lesen. Die Zeilenblockung ist jedoch inaktiviert.
- Verwenden der Klausel WITH RR USE AND KEEP UPDATE LOCKS oder WITH RS USE AND KEEP UPDATE LOCKS bei der Ausführung der Abfrage. Durch beide Klauseln wird sichergestellt, dass eine Sperre des Modus U aktiviert wird, wenn der Prozess A versucht, die Daten zu lesen, wobei Zeilenblockung zulässig ist.

Anmerkung: Es kann sinnvoll sein, einen Monitor zu definieren, der das Auftreten gegenseitiger Sperren aufzeichnet. Verwenden Sie zur Erstellung eines Monitors die SQL-Anweisung CREATE EVENT.

Bei der Erstellung einer Datenbank wird gleichzeitig auch ein detaillierter Ereignismonitor für gegenseitige Sperren erstellt. Wie bei jedem Monitor fällt für diesen Ereignismonitor etwas Systemaufwand an. Wenn der detaillierte Ereignismonitor für gegenseitige Sperren 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 Nachricht wird in das Protokoll mit Benachrichtigungen für die Systemverwaltung geschrieben, wenn die maximale Anzahl von Ausgabedateien erreicht ist. Durch das Entfernen der Ausgabedateien, die nicht mehr benötigt werden, kann der Ereignismonitor bei der nächsten Datenbankaktivierung erneut aktiviert werden.

In einer Systemumgebung mit zusammengeschlossenen Datenbanken, in der eine Anwendung auf Kurznamen zugreift, ist es möglich, dass die von der Anwendung angeforderten Daten aufgrund einer gegenseitigen Sperre an einer Datenquelle nicht verfügbar sind. In diesem Fall ist DB2® von den Einrichtungen zur Behandlung gegenseitiger Sperren an der Datenquelle abhängig. Wenn gegenseitige Sper-

ren über mehrere Datenquellen auftreten, ist DB2 für die Auflösung der gegenseitigen Sperren auf die Zeitlimitmechanismen der Datenquellen angewiesen.

Wenn Sie mehr Informationen über gegenseitige Sperren protokollieren wollen, setzen Sie den Konfigurationsparameter *notifylevel* des Datenbankmanagers auf den Wert 4. Das Benachrichtigungsprotokoll für die Systemverwaltung speichert Informationen zum Objekt, zum Sperrmodus und zur Anwendung, die die Sperre für das Objekt aktiviert hat. Der Name der aktuellen dynamischen SQL-Anweisung oder des statischen Pakets kann ebenfalls protokolliert werden. Die dynamische SQL-Anweisung wird nur bei Benachrichtigungsstufe (*notifylevel*) 4 protokolliert.

Zugehörige Konzepte:

- „Sperren und Steuerung des gemeinsamen Zugriffs" auf Seite 53
- „Gegenseitige Sperren zwischen Anwendungen" auf Seite 11

Zugehörige Tasks:

- „Korrigieren von Problemen der Sperreneskalation" auf Seite 64

Zugehörige Referenzen:

- „diaglevel - Aufzeichnungsebene bei Fehlerdiagnose" auf Seite 524
- „locktimeout - Zeitlimit für Sperren" auf Seite 432

Richtlinien für Sperren

Berücksichtigen Sie die folgenden Richtlinien, wenn Sie das Sperren im Hinblick auf den gemeinsamen Zugriff und die Datenintegrität optimieren:

- Erstellen Sie kleine Arbeitseinheiten mit häufigen Anweisungen COMMIT, um den gemeinsamen Zugriff auf Daten durch zahlreiche Benutzer zu fördern.

  Nehmen Sie Anweisungen COMMIT in die Anwendung an den Stellen auf, an denen sich die Anwendung logisch in einem Konsistenzzustand befindet, das heißt, wenn die geänderten Daten konsistent sind. Wenn eine Anweisung COMMIT abgesetzt wird, werden Sperren freigegeben, abgesehen von Tabellensperren, die mit der Klausel WITH HOLD deklarierten Cursorn zugeordnet sind.
- Geben Sie eine geeignete Isolationsstufe an.

  Sperren werden aktiviert, auch wenn die Anwendung Zeilen lediglich liest, so dass auch Arbeitseinheiten im Lesezugriff festgeschrieben (COMMIT) werden müssen. Der Grund dafür ist der, dass in den Isolationsstufen RR, RS und CS in Anwendungen mit Lesezugriff gemeinsame Sperren (Shared) aktiviert werden. Bei den Isolationsstufen RR und RS werden alle Sperren beibehalten, bis eine Anweisung COMMIT ausgeführt wird, so dass keine anderen Prozesse die gesperrten Daten aktualisieren können, es sei denn, Sie schließen den Cursor mit der Klausel WITH RELEASE. Außerdem werden auch Katalogsperren in Anwendungen mit dynamischem SQL unter der Isolationsstufe UR aktiviert.

  Der Datenbankmanager stellt sicher, dass die Anwendung keine noch nicht festgeschriebenen Daten (Zeilen, die von anderen Anwendungen aktualisiert, aber noch nicht mit der Anweisung COMMIT festgeschrieben wurden) abruft, sofern Sie nicht mit der Isolationsstufe UR (Uncommitted Read) arbeiten.
- Verwenden Sie die Anweisung LOCK TABLE in geeigneter Weise.

  Diese Anweisung sperrt eine ganze Tabelle. Es wird nur die in der Anweisung LOCK TABLE angegebene Tabelle gesperrt. Übergeordnete und abhängige Tabellen der angegebenen Tabelle werden nicht gesperrt. Sie müssen selbst entscheiden, ob das Sperren anderer Tabellen, auf die möglicherweise zugegriffen wird, erforderlich ist, um das gewünschte Ergebnis hinsichtlich des gemeinsamen

Zugriffs und der Leistung zu erzielen. Die Sperre wird erst freigegeben, wenn die Arbeitseinheit festgeschrieben oder rückgängig gemacht wurde.

LOCK TABLE IN SHARE MODE

Sie wollen auf Daten zugreifen, die *zeitlich konsistent* sind, d. h. Daten, die für eine Tabelle zu einem bestimmten Zeitpunkt aktuell sind. Wenn auf die Tabelle häufig zugegriffen wird, liegt die einzige Möglichkeit, die Tabelle in einem stabilen Zustand zu erhalten, darin, die Tabelle zu sperren. Die Anwendung soll zum Beispiel eine Momentaufnahme der Tabelle machen. Während der Zeit jedoch, die die Anwendung zur Verarbeitung einiger Zeilen der Tabelle benötigt, aktualisieren andere Anwendungen Zeilen, die die Anwendung noch nicht verarbeitet hat. Dies ist unter der Isolationsstufe RR (Wiederholtes Lesen) zulässig, jedoch nicht beabsichtigt.

Als alternative Methode kann Ihre Anwendung die Anweisung LOCK TABLE IN SHARE MODE absetzen: Es können keine Zeilen geändert werden, egal ob sie von Ihnen abgerufen wurden oder nicht. Anschließend kann die Anwendung die benötigte Anzahl von Zeilen abrufen und gleichzeitig davon ausgehen, dass die abgerufenen Zeilen nicht kurz vor dem Abrufen geändert wurden.

Nach Ausführung der Anweisung LOCK TABLE IN SHARE MODE können andere Benutzer Daten aus der Tabelle abrufen, jedoch keine Zeilen in der Tabelle aktualisieren, löschen oder einfügen.

LOCK TABLE IN EXCLUSIVE MODE

Sie beabsichtigen, einen großen Teil der Tabelle zu aktualisieren. Es ist weniger aufwendig und daher effektiver, alle anderen Benutzer vom Zugriff auf die Tabelle auszuschließen, als jede Zeile zur Aktualisierung zu sperren und anschließend die Sperren wieder freizugeben, wenn alle Änderungen festgeschrieben sind.

Durch LOCK TABLE IN EXCLUSIVE MODE werden alle anderen Benutzer ausgeschlossen, d. h. keine anderen Anwendungen können auf die Tabelle zugreifen, außer wenn es sich um Anwendungen mit der Isolationsstufe UR (nicht festgeschriebenes Lesen) handelt.

- Verwenden Sie Anweisungen ALTER TABLE in Anwendungen.

  Die Anweisung ALTER TABLE mit dem Parameter LOCKSIZE stellt eine Alternative zur Anweisung LOCK TABLE dar. Über den Parameter LOCKSIZE können Sie eine Sperrengranularität für den nächsten Tabellenzugriff angeben. Gültige Werte sind ROW für Zeilensperre oder TABLE für Tabellensperre.

  Die Auswahl von Zeilensperren (ROW) entspricht der Auswahl der Standardsperrengranularität bei der Erstellung einer Tabelle. Die Auswahl von Tabellensperren (TABLE) verbessert eventuell die Leistung von Abfragen durch die dadurch verringerte Anzahl benötigter Sperren. Allerdings könnte der gemeinsame Zugriff eingeschränkt werden, weil alle Sperren für die gesamte Tabelle aktiviert werden. Keine der Auswahlmöglichkeiten verhindert eine normale Sperreneskalation.

- Schließen Sie Cursor, um die von ihnen aktivierten Sperren freizugeben.

  Wenn Sie einen Cursor mit der Anweisung CLOSE CURSOR schließen, die die Klausel WITH RELEASE enthält, versucht der Datenbankmanager alle vorhandenen Lesesperren freizugeben, die für diesen Cursor aktiviert sind. Tabellenlesesperren sind IS-, S- und U-Tabellensperren. Zeilenlesesperren sind S-, NS- und U-Zeilensperren. Blocklesesperren sind IS-, S- und U-Blocksperren.

  Die Klausel WITH RELEASE hat keine Auswirkung auf Cursor, die unter den Isolationsstufen CS oder UR verwendet werden. Wenn die Klausel WITH

RELEASE für Cursor angegeben wird, die unter der Isolationsstufe RS oder RR verwendet werden, hebt sie einige Merkmale dieser Isolationsstufen auf. Dies heißt im Einzelnen, dass beim einem RS-Cursor *nichtwiederholbare Lesevorgänge* und bei einem RR-Cursor *nichtwiederholbare Lesevorgänge* oder *Phantomzeilen* auftreten können.

Wird ein Cursor, der ursprünglich ein RR- oder RS-Cursor ist, nachdem er mit der Klausel WITH RELEASE geschlossen wurde, erneut geöffnet, werden neue Lesesperren aktiviert.

In CLI-Anwendungen kann das DB2®-CLI-Verbindungsattribut SQL_ATTR_CLOSE_BEHAVIOR verwendet werden, um die gleichen Ergebnisse wie mit CLOSE CURSOR WITH RELEASE zu erzielen.

- Wenn Sie in einer partitionierten Datenbank Konfigurationsparameter ändern, die sich auf Sperren auswirken, stellen Sie sicher, dass die Änderungen in allen Partitionen durchgeführt werden.

Zugehörige Konzepte:

- „Sperren und Steuerung des gemeinsamen Zugriffs" auf Seite 53
- „Sperrenattribute" auf Seite 55
- „Sperren und Leistung" auf Seite 57
- „Faktoren mit Auswirkungen auf Sperren" auf Seite 79

Korrigieren von Problemen der Sperreneskalation

Der Datenbankmanager kann Sperren automatisch von Zeilen- oder Blockebene auf Tabellenebene eskalieren (hochstufen). Der Datenbankkonfigurationsparameter *maxlocks* gibt an, wann die Sperreneskalation ausgelöst wird. Die Tabelle, für die die Sperre aktiviert wird, die die Sperreneskalation auslöst, kann selbst unberührt bleiben. Sperren werden zuerst für die Tabelle mit den meisten Sperren eskaliert, angefangen bei Tabellen, für die LOB- und LONG VARCHAR-Deskriptoren gesperrt sind, dann für die Tabelle mit der nächsthöchsten Anzahl von Sperren usw., bis die Anzahl der aktiven Sperren auf ungefähr die Hälfte des durch *maxlocks* definierten Werts gesenkt wurde.

In einer adäquat entwickelten Datenbank kommt eine Sperreneskalation nur selten vor. Wenn die Sperreneskalation jedoch den gemeinsamen Zugriff auf ein nicht akzeptables Maß reduziert, müssen Sie das Problem analysieren und entscheiden, wie es zu lösen ist.

Voraussetzungen:

Stellen Sie sicher, dass Informationen zur Sperreneskalation aufgezeichnet werden. Setzen Sie den Konfigurationsparameter *notifylevel* (Benachrichtigungsstufe) des Datenbankmanagers auf den Wert 3 (Standardwert) oder 4. Wenn *notifylevel* den Wert 2 hat, wird nur der SQLCODE-Wert des Fehlers protokolliert. Wenn bei der Benachrichtigungsstufe 3 oder 4 eine Sperreneskalation fehlschlägt, werden Informationen zum SQLCODE-Wert des Fehlers sowie zur Tabelle aufgezeichnet, für die die Eskalation fehlgeschlagen ist. Die aktuelle SQL-Anweisung wird nur protokolliert, wenn es sich um eine gerade ausgeführte dynamische SQL-Anweisung handelt und der Parameter *notifylevel* auf den Wert 4 gesetzt ist.

Vorgehensweise:

Befolgen Sie zur Diagnose der Ursache für nicht akzeptable Sperreneskalationen sowie zur Anwendung von Abhilfemaßnahmen die folgenden Schritte:

1. Analysieren Sie das Benachrichtigungsprotokoll für die Systemverwaltung für alle Tabellen, für die Sperren eskaliert werden. Diese Protokolldatei enthält die folgenden Informationen:
   - Die Anzahl der momentan aktiven Sperren.
   - Die Anzahl der bis zum Ende der Sperreneskalation erforderlichen Sperren.
   - Die Tabellenkennung und der Tabellenname jeder eskalierten Tabelle.
   - Die Anzahl der momentan aktiven Sperren für andere Objekte als Tabellen.
   - Die neue Sperre auf Tabellenebene, die als Teil der Eskalation aktiviert werden soll. Dabei handelt es sich in der Regel um eine Sperre im Modus „S" (Share) oder „X" (eXclusive).
   - Der interne Rückkehrcode für das Ergebnis des Aktivierens der neuen Tabellensperrenebene.
2. Verwenden Sie die Informationen im Benachrichtigungsprotokoll für die Systemverwaltung, um zu entscheiden, wie das Eskalationsproblem zu lösen ist. Ziehen Sie die folgenden Möglichkeiten in Betracht:
   - Erhöhen der Anzahl der global zulässigen Sperren durch Heraufsetzen des Werts des Parameters *maxlocks*, des Parameters *locklist* oder beider Parameter in der Konfigurationsdatei der Datenbank. In einer partitionierten Datenbank führen Sie diese Änderungen in allen Partitionen aus.

     Sie könnten diese Methode wählen, wenn der gleichzeitige Zugriff auf die Tabelle durch andere Prozesse von höchster Wichtigkeit ist. Jedoch kann der Systemaufwand, der mit der Aktivierung von Sperren auf Datensatzebene verbunden ist, zu größeren Verzögerungen für andere Prozesse führen, als durch den gleichzeitigen Zugriff auf eine Tabelle an Zeit eingespart werden kann.
   - Anpassen des Prozesses bzw. der Prozesse, die die Eskalation verursacht haben. Für diese Prozesse könnten Sie die Anweisung LOCK TABLE explizit absetzen.
   - Ändern der Isolationsstufe. Beachten Sie, dass diese Maßnahme jedoch zu einer Verringerung des gemeinsamen Zugriffs führen kann.
   - Erhöhen der Häufigkeit von COMMIT-Operationen zur Verringerung der Anzahl von Sperren, die zu einem gegebenen Zeitpunkt aktiv sind.
   - Verwenden häufiger COMMIT-Anweisungen für Transaktionen, die LONG VARCHAR-Daten und verschiedene Arten von LOB-Daten anfordern. Diese Art von Daten wird zwar erst vom Datenträger gelesen, wenn die Ergebnismenge im Speicher erstellt wird, jedoch wird der Deskriptor bereits beim ersten Verweis auf die Daten gesperrt. Infolgedessen werden möglicherweise wesentlich mehr Sperren aktiviert als für Zeilen, die gängigere Arten von Daten enthalten.

Zugehörige Referenzen:

- „maxlocks - Maximale Anzahl Sperren pro Anwendung" auf Seite 433
- „diaglevel - Aufzeichnungsebene bei Fehlerdiagnose" auf Seite 524

Auswerten nicht festgeschriebener Daten durch Sperrenverzögerung

Zur Verbesserung des gemeinsamen Zugriffs lässt DB2® jetzt die Verzögerung von Zeilensperren für Suchen mit der Isolationsstufe CS oder RS in einigen Fällen zu, bis ein Datensatz festgestellt wird, der die Vergleichselemente einer Abfrage erfüllt. Standardmäßig sperrt DB2 beim Sperren von Zeilen während einer Tabellen- oder Indexsuche jede Zeile, die durchsucht wird, bevor festgestellt wird, ob die Zeile den Vergleichselementen der Abfrage entspricht. Zur Verbesserung des gemeinsamen Zugriffs durch Suchoperationen ist es möglich, die Zeilensperre zu verzögern, bis festgestellt ist, dass eine Zeile einer Abfrage entspricht.

Zur Nutzung dieser Funktion aktivieren Sie die Registriervariable DB2_EVALUNCOMMITTED.

Wenn diese Variable aktiviert ist, kann eine Auswertung von Vergleichselementen für nicht festgeschriebene Daten stattfinden. Das bedeutet, dass eine Zeile, die eine nicht festgeschriebene Aktualisierung enthält, die Abfrage vielleicht nicht erfüllt, während sie der Abfrage möglicherweise entsprechen würde, wenn mit der Auswertung der Vergleichselemente gewartet würde, bis die aktualisierte Transaktion beendet ist. Darüber hinaus werden nicht festgeschriebene gelöschte Zeilen bei Tabellensuchen übersprungen. DB2 überspringt gelöschte Schlüssel beim Durchsuchen von Indizes des Typs 2, wenn die Registriervariable DB2_SKIPDELETED aktiviert ist.

Diese Registriervariableneinstellungen gelten bei der Kompilierung für dynamisches SQL und beim Binden für statisches SQL. Dies bedeutet, dass selbst wenn die Registriervariable bei der Ausführung aktiviert ist, die Sperrenvermeidungsstrategie nicht angewendet wird, wenn die Registriervariable DB2_EVALUNCOMMITTED beim Binden nicht aktiviert war. Wenn die Registriervariable beim Binden aktiviert, bei der Ausführung jedoch nicht aktiviert ist, wird die Sperrenvermeidungsstrategie trotzdem angewendet. Für statisches SQL gilt beim erneuten Binden eines Pakets die Einstellung der Registriervariablen zum Zeitpunkt des Bindens. Ein implizites erneutes Binden von statischem SQL verwendet die aktuelle Einstellung der Registriervariablen DB2_EVALUNCOMMITTED.

Anwendbarkeit der Auswertung nicht festgeschriebener Daten für verschiedene Zugriffspläne

Tabelle 4. Zugriff nur über Satz-ID-Index

Vergleichselemente	Nicht festgeschriebene Daten auswerten
Keine	Nein
Als Suchargument verwendbare Vergleichselemente	Ja

Tabelle 5. Reiner Datenzugriff (relational oder mit verzögerter Satz-ID-Liste)

Vergleichselemente	Nicht festgeschriebene Daten auswerten
Keine	Nein
Als Suchargument verwendbare Vergleichselemente	Ja

Tabelle 6. Satz-ID-Index und Datenzugriff

Vergleichselemente		Nicht festgeschriebene Daten auswerten	
Index	**Daten**	**Indexzugriff**	**Datenzugriff**
Keine	Keine	Nein	Nein
Keine	Als Suchargument verwendbare Vergleichselemente	Nein	Nein
Als Suchargument verwendbare Vergleichselemente	Keine	Ja	Nein
Als Suchargument verwendbare Vergleichselemente	Als Suchargument verwendbare Vergleichselemente	Ja	Nein

Tabelle 7. Blockindex und Datenzugriff

Vergleichselemente		Nicht festgeschriebene Daten auswerten	
Index	**Daten**	**Indexzugriff**	**Datenzugriff**
Keine	Keine	Nein	Nein
Keine	Als Suchargument verwendbare Vergleichselemente	Nein	Ja
Als Suchargument verwendbare Vergleichselemente	Keine	Ja	Nein
Als Suchargument verwendbare Vergleichselemente	Als Suchargument verwendbare Vergleichselemente	Ja	Ja

Beispiel

Das folgende Beispiel zeigt einen Vergleich der Standardfunktionsweise von Sperren und der neuen Funktionsweise mit Auswertung nicht festgeschriebener Daten.

Die folgende Tabelle ist die Tabelle ORG aus der Beispieldatenbank SAMPLE.

```
DEPTNUMB DEPTNAME       MANAGER DIVISION   LOCATION
-------- -------------- ------- ---------- -------------
      10 Head Office        160 Corporate  New York
      15 New England         50 Eastern    Boston
      20 Mid Atlantic        10 Eastern    Washington
      38 South Atlantic      30 Eastern    Atlanta
      42 Great Lakes        100 Midwest    Chicago
      51 Plains             140 Midwest    Dallas
      66 Pacific            270 Western    San Francisco
      84 Mountain           290 Western    Denver
```

Die folgenden Transaktionen werden an dieser Tabelle mit der Standardisolationsstufe der Cursorstabilität (CS) ausgeführt.

Tabelle 8. Transaktionen an der Tabelle ORG mit der Isolationsstufe CS

SITZUNG 1	SITZUNG 2
`connect to SAMPLE`	`connect to SAMPLE`
`+c update org set  deptnumb=5 where manager=160`	
	`select * from org where deptnumb >= 10`

Die nicht festgeschriebene UPDATE-Operation in Sitzung 1 aktiviert eine exklusive Satzsperre für die erste Zeile in der Tabelle, welche die SELECT-Abfrage in Sitzung 2 daran hindert, zurückzukehren, obwohl die in Sitzung 1 zu aktualisierende Zeile momentan die Abfrage in Sitzung 2 nicht erfüllt. Das liegt daran, dass die Isolationsstufe CS festlegt, dass jede Zeile, auf die durch eine Abfrage zugegriffen wird, gesperrt werden muss, während sich der Cursor auf dieser Zeile befindet. Sitzung 2 kann solange keine Sperre für die erste Zeile aktivieren, bis Sitzung 1 ihre Sperre freigibt.

Beim Durchsuchen der Tabelle kann das Warten auf die Sperre in Sitzung 2 durch die Verwendung der Funktion zur Auswertung nicht festgeschriebener Daten vermieden werden. Dabei wird zunächst das Vergleichselement ausgewertet und anschließend eine Sperre für die Zeile aktiviert, um das Vergleichselement endgültig auszuwerten. In diesem Fall würde die Abfrage in Sitzung 2 nicht versuchen, die erste Zeile in der Tabelle zu sperren, so dass der gemeinsame Zugriff durch Anwendungen verbessert wird. Beachten Sie, dass dies auch bedeuten würde, dass die Auswertung des Vergleichselements in Sitzung 2 im Hinblick auf den nicht festgeschriebenen Wert mit deptnumb=5 in Sitzung 1 erfolgen würde. Die Abfrage in Sitzung 2 würde die erste Zeile aus ihrer Ergebnismenge herauslassen, obwohl ein Rückgängigmachen der UPDATE-Operation in Sitzung 1 die Abfrage in Sitzung 2 erfüllen würde.

Wenn die Reihenfolge der Operationen umgekehrt wäre, ergäbe sich mit der Funktion zur Auswertung nicht festgeschriebener Daten trotzdem eine Verbesserung für den gemeinsamen Zugriff. Bei der Standardfunktionsweise von Sperren würde Sitzung 2 zunächst eine Zeilensperre anfordern, die die Ausführung der Aktualisierung mit Suche in Sitzung 1 verhindern würde, obwohl die Aktualisierung in Sitzung 1 die von der Abfrage in Sitzung 2 gesperrte Zeile nicht ändern würde. Wenn die Aktualisierung mit Suche in Sitzung 1 versuchen würde, die Zeilen zunächst zu untersuchen und sie nur zu sperren, wenn sie die Vergleichselemente erfüllen, würde die Abfrage in Sitzung 1 nicht blockiert.

Einschränkungen

Die folgenden externen Einschränkungen gelten für diese neue Funktionalität:

- Die Registriervariable DB2_EVALUNCOMMITTED muss aktiviert sein.
- Die Isolationsstufe muss CS oder RS sein.
- Zeilensperren müssen auftreten.
- Als Suchargument verwendbare Vergleichselemente sind vorhanden.
- Die Auswertung nicht festgeschriebener Daten gilt nicht für Suchen in den Katalogtabellen.
- Für MDC-Tabellen können Blocksperren für eine Indexsuche verzögert werden. Bei Tabellensuchen werden Blocksperren jedoch nicht verzögert.
- Das verzögerte Sperren tritt nicht für eine Tabelle auf, die eine Inplace-Tabellenreorganisation ausführt.
- Das verzögerte Sperren findet nicht bei einer Indexsuche statt, wenn es sich um einen Index des Typs 1 handelt.
- Für Iscan-Fetch-Pläne werden Zeilensperren nicht bis zum Datenzugriff verzögert, sondern die Zeile wird beim Indexzugriff gesperrt, bevor auf die Zeile in der Tabelle zugegriffen wird.
- Gelöschte Zeilen werden bei Tabellensuchen bedingungslos übersprungen, während gelöschte Schlüssel von Indizes des Typs 2 nur übersprungen werden, wenn die Registriervariable DB2_SKIPDELETED aktiviert ist.

Sperrentypenkompatibilität

Die folgende Tabelle enthält Informationen zu den Umständen, in denen eine Sperrenanforderungen erfüllt werden kann, wenn ein anderer Prozess eine Sperre für dieselbe Ressource in einem bestimmten Status aktiviert oder angefordert hat. Ein **Nein** bedeutet, dass der Anforderer warten muss, bis alle inkompatiblen Sperren von anderen Prozessen inaktiviert werden. Beachten Sie, dass es zu einer Zeitlimitüberschreitung kommen kann, wenn ein Anforderer auf eine Sperre wartet. Ein **Ja** bedeutet, dass die Sperre aktiviert wird, sofern kein früherer Anforderer auf die Ressource wartet.

Tabelle 9. Kompatibilität der Sperrmodi

	Status der aktiven Sperre											
Angeforderter Status	**NONE**	**IN**	**IS**	**NS**	**S**	**IX**	**SIX**	**U**	**X**	**Z**	**NW**	**W**
NONE	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja
IN	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Nein	Ja	Ja
IS	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Nein	Nein	Nein	Nein
NS	Ja	Ja	Ja	Ja	Ja	Nein	Nein	Ja	Nein	Nein	Ja	Nein
S	Ja	Ja	Ja	Ja	Ja	Nein	Nein	Ja	Nein	Nein	Nein	Nein
IX	Ja	Ja	Ja	Nein	Nein	Ja	Nein	Nein	Nein	Nein	Nein	Nein
SIX	Ja	Ja	Ja	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein
U	Ja	Ja	Ja	Ja	Ja	Nein	Nein	Nein	Nein	Nein	Nein	Nein
X	Ja	Ja	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein
Z	Ja	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein
NW	Ja	Ja	Nein	Ja	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Ja
W	Ja	Ja	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Ja	Nein

Anmerkung:

I	Intent
N	NONE
NS	Next Key Share
S	Share
X	Exclusive
U	Update
Z	Super Exclusive
NW	Next Key Weak Exclusive
W	Weak Exclusive

Anmerkung:
- Ja - angeforderte Sperre sofort **erteilen**
- Nein - **warten**, bis aktive Sperre freigegeben oder Zeitlimit überschritten wird

Zugehörige Konzepte:
- „Sperren und Steuerung des gemeinsamen Zugriffs" auf Seite 53
- „Sperrenattribute" auf Seite 55
- „Sperren und Leistung" auf Seite 57

Zugehörige Referenzen:
- „Sperrmodi und Zugriffspfade für Standardtabellen" auf Seite 70
- „Sperren für Blockindexsuchen für MDC-Tabellen" auf Seite 77

Sperrmodi und Zugriffspfade für Standardtabellen

Dieser Abschnitt enthält Referenzinformationen zu Sperrmethoden für Standardtabellen für verschiedene Datenzugiffspläne. In den folgenden Tabellen werden die Typen von Sperren aufgelistet, die für Standardtabellen in der jeweiligen Isolationsstufe für verschiedene Zugriffspläne aktiviert werden. Jeder Eintrag besteht aus zwei Teilen: Tabellensperre und Zeilensperre. Ein Strich zeigt an, dass für eine bestimmte Stufe keine Sperrung erfolgt.

Anmerkungen:

1. In einer MDC-Umgebung (Multidimensional Clustering) wird eine zusätzliche Sperrstufe verwendet: der Block.
2. Sperrmodi können explizit mit einer Sperranforderungsklausel (LOCK REQUEST) in einer SELECT-Anweisung geändert werden.

Tabelle 10. Sperrmodi für Tabellensuchen

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche	UPDATE oder DELETE
Zugriffsmethode: *Tabellensuche ohne Vergleichselemente*					
RR	S/-	U/-	SIX/X	X/-	X/-
RS	IS/NS	IX/U	IX/X	IX/X	IX/X
CS	IS/NS	IX/U	IX/X	IX/X	IX/X
UR	IN/-	IX/U	IX/X	IX/X	IX/X
Zugriffsmethode: *Tabellensuche mit Vergleichselementen*					
RR	S/-	U/-	SIX/X	U/-	SIX/X
RS	IS/NS	IX/U	IX/X	IX/U	IX/X
CS	IS/NS	IX/U	IX/X	IX/U	IX/X
UR	IN/-	IX/U	IX/X	IX/U	IX/X

Anmerkung: Wenn in der Isolationsstufe UR mit IN-Sperren für Indizes des Typs 1 Vergleichselemente für INCLUDE-Spalten im Index angewendet werden, werden die Isolationsstufe auf CS erhöht und die Sperren in eine IS-Tabellensperre und in NS-Zeilensperren umgewandelt.

Tabelle 11. Sperrmodi für Satz-ID-Indexsuchen (RID)

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche	UPDATE oder DELETE
Zugriffsmethode: *Satz-ID-Indexsuche ohne Vergleichselemente*					
RR	S/-	IX/S	IX/X	X/-	X/-
RS	IS/NS	IX/U	IX/X	IX/X	IX/X
CS	IS/NS	IX/U	IX/X	IX/X	IX/X
UR	IN/-	IX/U	IX/X	IX/X	IX/X
Zugriffsmethode: *Satz-ID-Indexsuche mit einer zu findenden Zeile*					
RR	IS/S	IX/U	IX/X	IX/X	IX/X
RS	IS/NS	IX/U	IX/X	IX/X	IX/X
CS	IS/NS	IX/U	IX/X	IX/X	IX/X
UR	IN/-	IX/U	IX/X	IX/X	IX/X
Zugriffsmethode: *Indexsuche nur mit Start- und Stoppvergleichselementen*					

Tabelle 11. Sperrmodi für Satz-ID-Indexsuchen (RID) (Forts.)

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche	UPDATE oder DELETE
RR	IS/S	IX/S	IX/X	IX/X	IX/X
RS	IS/NS	IX/U	IX/X	IX/X	IX/X
CS	IS/NS	IX/U	IX/X	IX/X	IX/X
UR	IN/-	IX/U	IX/X	IX/X	IX/X
Zugriffsmethode: ***Indexsuche nur mit Index und anderen Vergleichselementen (sargs, resids)***					
RR	IS/S	IX/S	IX/X	IX/S	IX/X
RS	IS/NS	IX/U	IX/X	IX/U	IX/X
CS	IS/NS	IX/U	IX/X	IX/U	IX/X
UR	IN/-	IX/U	IX/X	IX/U	IX/X

Die folgende Tabelle zeigt die Sperrmodi für Fälle, in denen der Lesezugriff auf die Datenseiten verzögert erfolgt, um folgende Operationen an der Zeilenliste durchführen zu können:

- Eine weitere Qualifizierung mit Hilfe mehrerer Indizes
- Eine Sortierung für einen effizienten Vorablesezugriff

Tabelle 12. Sperrmodi für Indexsuchen mit verzögertem Zugriff auf Datenseiten

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche	UPDATE oder DELETE
Zugriffsmethode: ***Satz-ID-Indexsuche ohne Vergleichselemente***					
RR	IS/S	IX/S		X/-	
RS	IN/-	IN/-		IN/-	
CS	IN/-	IN/-		IN/-	
UR	IN/-	IN/-		IN/-	
Zugriffsmethode: ***Verzögerter Zugriff auf Datenseiten nach Satz-ID-Indexsuche ohne Vergleichselemente***					
RR	IN/-	IX/S	IX/X	X/-	X/-
RS	IS/NS	IX/U	IX/X	IX/X	IX/X
CS	IS/NS	IX/U	IX/X	IX/X	IX/X
UR	IN/-	IX/U	IX/X	IX/X	IX/X
Zugriffsmethode: ***Satz-ID-Indexsuche mit Vergleichselementen (sargs, resids)***					
RR	IS/S	IX/S		IX/S	
RS	IN/-	IN/-		IN/-	
CS	IN/-	IN/-		IN/-	
UR	IN/-	IN/-		IN/-	

Tabelle 12. Sperrmodi für Indexsuchen mit verzögertem Zugriff auf Datenseiten (Forts.)

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche	UPDATE oder DELETE
Zugriffsmethode: *Satz-ID-Indexsuche nur mit Start- und Stoppvergleichselementen*					
RR	IS/S	IX/S		IX/X	
RS	IN/-	IN/-		IN/-	
CS	IN/-	IN/-		IN/-	
UR	IN/-	IN/-		IN/-	
Zugriffsmethode: *Verzögerter Zugriff auf Datenseiten nach Satz-ID-Indexsuche nur mit Start- und Stoppvergleichselementen*					
RR	IN/-	IX/S	IX/X	IX/X	IX/X
RS	IS/NS	IX/U	IX/X	IX/U	IX/X
CS	IS/NS	IX/U	IX/X	IX/U	IX/X
UR	IS/-	IX/U	IX/X	IX/U	IX/X
Zugriffsmethode: *Verzögerter Zugriff auf Datenseiten nach Satz-ID-Indexsuche mit Vergleichselementen*					
RR	IN/-	IX/S	IX/X	IX/S	IX/X
RS	IS/NS	IX/U	IX/X	IX/U	IX/X
CS	IS/NS	IX/U	IX/X	IX/U	IX/X
UR	IN/-	IX/U	IX/X	IX/U	IX/X

Zugehörige Konzepte:

- „Sperrenattribute" auf Seite 55
- „Sperren und Leistung" auf Seite 57

Zugehörige Referenzen:

- „Sperrentypenkompatibilität" auf Seite 69
- „Sperrmodi für Tabellen- und Satz-ID-Indexsuchen für MDC-Tabellen" auf Seite 73
- „Sperren für Blockindexsuchen für MDC-Tabellen" auf Seite 77

Sperrmodi für Tabellen- und Satz-ID-Indexsuchen für MDC-Tabellen

In einer MDC-Umgebung (Multidimensional Clustering) wird eine zusätzliche Sperrstufe verwendet: der Block. In den folgenden Tabellen werden die Typen von Sperren aufgelistet, die in der jeweiligen Isolationsstufe für verschiedene Zugriffspläne aktiviert werden. Jeder Eintrag besteht aus drei Teilen: Tabellensperre, Blocksperre und Zeilensperre. Ein Strich zeigt an, dass für eine bestimmte Stufe keine Sperrung erfolgt.

Anmerkung: Sperrmodi können explizit mit einer Sperranforderungsklausel (LOCK REQUEST) in einer SELECT-Anweisung geändert werden.

Tabelle 13. Sperrmodi für Tabellensuchen

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
Zugriffsmethode: *Tabellensuche ohne Vergleichselemente*					
RR	S/-/-	U/-/-	SIX/IX/X	X/-/-	X/-/-
RS	IS/IS/NS	IX/IX/U	IX/IX/U	IX/X/-	IX/I/-
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-
Zugriffsmethode: *Tabellensuche mit Vergleichselementen nur für Dimensionsspalten*					
RR	S/-/-	U/-/-	SIX/IX/X	U/-/-	SIX/X/-
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/U/-	X/X/-
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/U/-	X/X/-
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/U/-	X/X/-
Zugriffsmethode: *Tabellensuche mit anderen Vergleichselementen (sargs, resids)*					
RR	S/-/-	U/-/-	SIX/IX/X	U/-/-	SIX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

Die beiden folgenden Tabellen zeigen Sperrmodi für Satz-ID-Indizes für MDC-Tabellen.

Tabelle 14. Sperrmodi für Satz-ID-Indexsuchen (RID)

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
Zugriffsmethode: *Satz-ID-Indexsuche ohne Vergleichselemente*					
RR	S/-/-	IX/IX/S	IX/IX/X	X/-/-	X/-/-
RS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/X	X/X/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/X	X/X/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	X/X/X	X/X/X
Zugriffsmethode: *Satz-ID-Indexsuche mit einer zu findenden Zeile*					
RR	IS/IS/S	IX/IX/U	IX/IX/X	X/X/X	X/X/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/X	X/X/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/X	X/X/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	X/X/X	X/X/X
Zugriffsmethode: *Satz-ID-Indexsuche nur mit Start- und Stoppvergleichselementen*					

Tabelle 14. Sperrmodi für Satz-ID-Indexsuchen (RID) (Forts.)

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	IS/IS/S	IX/IX/S	IX/IX/X	IX/IX/X	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X
Zugriffsmethode: *Indexsuche mit Indexvergleichselementen*					
RR	IS/S/S	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
Zugriffsmethode: *Indexsuche mit anderen Vergleichselementen (sargs, resids)*					
RR	IS/S/S	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

Anmerkung: In der folgenden Tabelle, die Sperrmodi für Satz-ID-Indexsuchen für verzögerte Datenseitenzugriffe zeigt, werden die Isolationsstufe auf CS erhöht und die Sperren in eine IS-Tabellensperre, eine IS-Blocksperre und NS-Zeilensperren hochgestuft, wenn die Isolationsstufe UR mit einer IN-Sperre für Indizes des Typs 1 vorliegt oder wenn es Vergleichselemente für INCLUDE-Spalten im Index gibt.

Tabelle 15. Sperrmodi für Satz-ID-Indexsuchen mit verzögertem Zugriff auf Datenseiten

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
Zugriffsmethode: *Satz-ID-Indexsuche ohne Vergleichselemente*					
RR	IS/S/S	IX/IX/S		X/-/-	
RS	IN/IN/-	IN/IN/-		IN/IN/-	
CS	IN/IN/-	IN/IN/-		IN/IN/-	
UR	IN/IN/-	IN/IN/-		IN/IN/-	
Zugriffsmethode: *Verzögerter Zugriff auf Datenseiten nach Satz-ID-Indexsuche ohne Vergleichselemente*					
RR	IN/IN/-	IX/IX/S	IX/IX/X	X/-/-	X/-/-
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X

Tabelle 15. Sperrmodi für Satz-ID-Indexsuchen mit verzögertem Zugriff auf Datenseiten (Forts.)

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
Zugriffsmethode: *Satz-ID-Indexsuche mit Vergleichselementen (sargs, resids)*					
RR	IS/S/-	IX/IX/S		IX/IX/S	
RS	IN/IN/-	IN/IN/-		IN/IN/-	
CS	IN/IN/-	IN/IN/-		IN/IN/-	
UR	IN/IN/-	IN/IN/-		IN/IN/-	
Zugriffsmethode: *Verzögerter Zugriff auf Datenseiten nach Satz-ID-Indexsuche mit Vergleichselementen (sargs, resids)*					
RR	IN/IN/-	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
Zugriffsmethode: *Satz-ID-Indexsuche nur mit Start- und Stoppvergleichselementen*					
RR	IS/IS/S	IX/IX/S		IX/IX/X	
RS	IN/IN/-	IN/IN/-		IN/IN/-	
CS	IN/IN/-	IN/IN/-		IN/IN/-	
UR	IN/IN/-	IN/IN/-		IN/IN/-	
Zugriffsmethode: *Verzögerter Zugriff auf Datenseiten nach Satz-ID-Indexsuche nur mit Start- und Stoppvergleichselementen*					
RR	IN/IN/-	IX/IX/S	IX/IX/X	IX/IX/X	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IS/-/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

Zugehörige Konzepte:

- „Sperren und Steuerung des gemeinsamen Zugriffs" auf Seite 53
- „Sperrenattribute" auf Seite 55
- „Sperren und Leistung" auf Seite 57

Zugehörige Referenzen:

- „Sperrentypenkompatibilität" auf Seite 69
- „Sperrmodi und Zugriffspfade für Standardtabellen" auf Seite 70
- „Sperren für Blockindexsuchen für MDC-Tabellen" auf Seite 77

Sperren für Blockindexsuchen für MDC-Tabellen

In den folgenden Tabellen werden die Typen von Sperren aufgelistet, die in der jeweiligen Isolationsstufe für verschiedene Zugriffspläne aktiviert werden. Jeder Eintrag besteht aus drei Teilen: Tabellensperre, Blocksperre und Zeilensperre. Ein Strich zeigt an, dass für eine bestimmte Stufe keine Sperrung erfolgt.

Anmerkung: Sperrmodi können explizit mit einer Sperranforderungsklausel (LOCK REQUEST) in einer SELECT-Anweisung geändert werden.

Tabelle 16. Sperrmodi für Indexsuchen

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
Zugriffsmethode: *Ohne Vergleichselemente*					
RR	S/--/--	IX/IX/S	IX/IX/X	X/--/--	X/--/--
RS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/--	X/X/--
CS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/--	X/X/--
UR	IN/IN/-	IX/IX/U	IX/IX/X	X/X/--	X/X/--
Zugriffsmethode: *Nur mit Dimensionsvergleichselementen*					
RR	IS/-/-	IX/IX/S	IX/IX/X	X/-/-	X/-/-
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-
Zugriffsmethode: *Nur mit Start- und Stoppvergleichselementen für Dimensionen*					
RR	IS/S/-	IX/IX/S	IX/IX/S	IX/IX/S	IX/IX/S
RS	IX/IX/S	IX/IX/U	IX/IX/X	IX/IX/-	IX/IX/-
CS	IX/IX/S	IX/IX/U	IX/IX/X	IX/IX/-	IX/IX/-
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/-	IX/IX/-
Zugriffsmethode: *Indexsuche mit Vergleichselementen*					
RR	IS/S/-	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

Die folgende Tabelle listet Sperrmodi für Blockindexsuchen mit verzögertem Zugriff auf Datenseiten auf:

Tabelle 17. Sperrmodi für Blockindexsuchen mit verzögertem Zugriff auf Datenseiten

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
Zugriffsmethode: *Blockindexsuche ohne Vergleichselemente*					
RR	IS/S/--	IX/IX/S		X/--/--	
RS	IN/IN/--	IN/IN/--		IN/IN/--	
CS	IN/IN/--	IN/IN/--		IN/IN/--	
UR	IN/IN/--	IN/IN/--		IN/IN/--	
Zugriffsmethode: *Verzögerter Zugriff auf Datenseiten nach Blockindexsuche ohne Vergleichselemente*					
RR	IN/IN/--	IX/IX/S	IX/IX/X	X/--/--	X/--/--
RS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/--	X/X/--
CS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/--	X/X/--
UR	IN/IN/--	IX/IX/U	IX/IX/X	X/X/--	X/X/--
Zugriffsmethode: *Blockindexsuche nur mit Dimensionsvergleichselementen*					
RR	IS/S/--	IX/IX/--		IX/S/--	
RS	IS/IS/NS	IX/--/--		IX/--/--	
CS	IS/IS/NS	IX/--/--		IX/--/--	
UR	IN/IN/--	IX/--/--		IX/--/--	
Zugriffsmethode: *Verzögerter Zugriff auf Datenseiten nach Blockindexsuche nur mit Dimensionsvergleichselementen*					
RR	IN/IN/--	IX/IX/S	IX/IX/X	IX/S/--	IX/X/--
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/U/--	IX/X/--
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/U/--	IX/X/--
UR	IN/IN/--	IX/IX/U	IX/IX/X	IX/U/--	IX/X/--
Zugriffsmethode: *Blockindexsuche nur mit Start- und Stoppvergleichselementen*					
RR	IS/S/--	IX/IX/--		IX/X/--	
RS	IN/IN/--	IN/IN/--		IN/IN/--	
CS	IN/IN/--	IN/IN/--		IN/IN/--	
UR	IN/IN/--	IN/IN/--		IN/IN/--	
Zugriffsmethode: *Verzögerter Zugriff auf Datenseiten nach Blockindexsuche nur mit Start- und Stoppvergleichselementen*					
RR	IN/IN/--	IX/IX/X		IX/X/--	
RS	IS/IS/NS	IN/IN/--		IN/IN/--	
CS	IS/IS/NS	IN/IN/--		IN/IN/--	
UR	IS/--/--	IN/IN/--		IN/IN/--	
Zugriffsmethode: *Blockindexsuche mit anderen Vergleichselementen (sargs, resids)*					
RR	IS/S/--	IX/IX/--		IX/IX/--	
RS	IN/IN/--	IN/IN/--		IN/IN/--	

Tabelle 17. Sperrmodi für Blockindexsuchen mit verzögertem Zugriff auf Datenseiten (Forts.)

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	**WHERE CURRENT OF**	**Suche od. DELETE**	**UPDATE**
CS	IN/IN/--	IN/IN/--		IN/IN/--	
UR	IN/IN/--	IN/IN/--		IN/IN/--	
Zugriffsmethode: ***Verzögerter Zugriff auf Datenseiten nach Blockindexsuche mit anderen Vergleichselementen (sargs, resids)***					
RR	IN/IN/--	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/--	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

Zugehörige Konzepte:

- „Sperren und Leistung" auf Seite 57

Zugehörige Referenzen:

- „Sperrentypenkompatibilität" auf Seite 69
- „Sperrmodi und Zugriffspfade für Standardtabellen" auf Seite 70
- „Sperrmodi für Tabellen- und Satz-ID-Indexsuchen für MDC-Tabellen" auf Seite 73

Faktoren mit Auswirkungen auf Sperren

Die folgenden Faktoren beeinflussen den Modus und die Granularität von Sperren des Datenbankmanagers:

- Die Art der Verarbeitung, die von der Anwendung ausgeführt wird
- Die Datenzugriffsmethode
- Der Typ der Indizes: 1 oder 2
- Verschiedene Konfigurationsparameter

Zugehörige Konzepte:

- „Sperren und Steuerung des gemeinsamen Zugriffs" auf Seite 53
- „Sperrenattribute" auf Seite 55
- „Sperren und Leistung" auf Seite 57
- „Richtlinien für Sperren" auf Seite 62
- „Indexbereinigung und Indexpflege" auf Seite 296
- „Sperren und Arten der Anwendungsverarbeitung" auf Seite 80
- „Sperren und Datenzugriffsmethoden" auf Seite 81
- „Indextypen und Sperren des nächsten Schlüssels" auf Seite 82

Faktoren mit Auswirkungen auf Sperren

Sperren und Arten der Anwendungsverarbeitung

Zum Zweck der Bestimmung von Sperrenattributen kann die Verarbeitung durch Anwendungen einem der folgenden Typen zugeordnet werden:

- Lesezugriff

  Dieser Typ umfasst alle SELECT-Anweisungen, die von sich aus nur Lesezugriff haben, eine explizite Klausel FOR READ ONLY enthalten oder mehrdeutig sind, jedoch vom SQL-Compiler aufgrund der im Befehl PREP oder BIND angegebenen Option BLOCKING als Anweisungen mit Lesezugriff eingestuft werden. Für diesen Verarbeitungstyp sind nur S-Sperren (S, NS oder IS) erforderlich.
- Änderungsabsicht

  Dieser Typ umfasst alle SELECT-Anweisungen mit der Klausel FOR UPDATE, der Klausel USE AND KEEP UPDATE LOCKS, der Klausel USE AND KEEP EXCLUSIVE LOCKS bzw. Anweisungen, die der SQL-Compiler als mehrdeutige Anweisung interpretiert, um implizit anzunehmen, dass eine Änderung beabsichtigt wird. Für diesen Typ werden Share- und Update-Sperren (S, U und X für Zeilen; IX, U, X und S für Blöcke; IX, U und X für Tabellen) verwendet.
- Änderung

  Dieser Typ umfasst Anweisungen mit UPDATE, INSERT und DELETE, aber keine Anweisungen mit UPDATE WHERE CURRENT OF oder DELETE WHERE CURRENT OF. Für diesen Typ sind exklusive Sperren (X oder IX) erforderlich.
- Cursorgesteuert

  Dieser Typ umfasst Anweisungen mit UPDATE WHERE CURRENT OF und DELETE WHERE CURRENT OF. Für diesen Typ sind ebenfalls Sperren des exklusiven Modus (X oder IX) erforderlich.

Eine Anweisung, die an einer Zieltabelle Einfüge-, Aktualisierungs- oder Löschoperationen (INSERT, UPDATE oder DELETE) auf der Grundlage des Ergebnisses einer Unterauswahl vornimmt, führt zwei Typen der Verarbeitung aus. Die Regeln für Lesezugriffsverarbeitung bestimmen die Sperren für die Tabellen, die in der Anweisung der Unterauswahl zurückgegeben werden. Die Regeln für die Änderungsverarbeitung bestimmen die Sperren für die Zieltabelle.

Zugehörige Konzepte:

- „Sperren und Steuerung des gemeinsamen Zugriffs" auf Seite 53
- „Sperrenattribute" auf Seite 55
- „Sperren und Leistung" auf Seite 57
- „Richtlinien für Sperren" auf Seite 62
- „Gegenseitige Sperren zwischen Anwendungen" auf Seite 11
- „Sperren und Datenzugriffsmethoden" auf Seite 81
- „Indextypen und Sperren des nächsten Schlüssels" auf Seite 82

Zugehörige Tasks:

- „Korrigieren von Problemen der Sperreneskalation" auf Seite 64

Zugehörige Referenzen:

- „Sperrentypenkompatibilität" auf Seite 69

Sperren und Datenzugriffsmethoden

Ein *Zugriffsplan* ist die Methode, die das Optimierungsprogramm zum Abrufen von Daten aus einer bestimmten Tabelle auswählt. Der Zugriffsplan kann erhebliche Auswirkungen auf Sperrmodi haben. Wenn zum Beispiel eine Indexsuche zum Auffinden einer bestimmten Zeile verwendet wird, wählt das Optimierungsprogramm wahrscheinlich Sperren auf Zeilenebene (IS) für die Tabelle aus. Wenn die Tabelle EMPLOYEE zum Beispiel einen Index für die Spalte EMPNO hat, könnte ein Zugriff über einen Index ausgeführt werden, um Informationen zu einem einzelnen Mitarbeiter mit Hilfe einer Anweisung auszuwählen, die die folgende SELECT-Klausel enthält:

```
SELECT *
  FROM EMPLOYEE
  WHERE EMPNO = '000310';
```

Wenn kein Index verwendet wird, muss die gesamte Tabelle der Reihe nach durchsucht werden, um die ausgewählten Zeilen zu finden. Dafür ist möglicherweise nur eine Sperre auf Tabellenebene (S) erforderlich. Wenn zum Beispiel kein Index für die Spalte SEX vorhanden ist, könnte eine Tabellensuche verwendet werden, um alle männlichen Mitarbeiter mit einer Anweisung auszuwählen, die folgende SELECT-Klausel enthält:

```
SELECT *
  FROM EMPLOYEE
  WHERE SEX = 'M';
```

Anmerkung: Bei der cursorgesteuerten Verarbeitung wird der Sperrmodus des zugrunde liegenden Cursors verwendet, bis die Anwendung eine Zeile findet, die zu aktualisieren oder zu löschen ist. Für diesen Verarbeitungstyp wird unabhängig vom Sperrmodus eines Cursors immer eine Sperre des Modus Exclusive aktiviert, um die Aktualisierung oder Löschung durchzuführen.

Das Sperren in Bereichsclustertabellen funktioniert etwas anders als beim Sperren von Standardschlüsseln und nächsten Schlüsseln. Beim Zugriff auf einen Bereich von Zeilen ineiner Bereichsclustertabelle werden alle Zeilen in diesem Bereich gesperrt, selbst wenn einige dieser Zeilen leer sind. Beim Sperren von Standardschlüsseln oder nächsten Schlüsseln werden nur Zeilen mit vorhandenen Datensätzen gesperrt.

Referenztabellen enthalten detaillierte Informationen darüber, welche Sperren für welche Art von Zugriffsplan aktiviert werden.

Beim verzögerten Zugriff auf die Datenseiten erfolgt der Zugriff auf die Zeile in zwei Schritten, wodurch komplexere Sperrsituationen auftreten. Die Zeitpunkte der Aktivierung von Sperren und die Dauer der Sperren hängen von der Isolationsstufe ab. Da bei der Isolationsstufe RR (*Wiederholtes Lesen*) alle Sperren bis zum Ende der Transaktion beibehalten werden, bleiben die im ersten Schritt aktivierten Sperren bestehen, und im zweiten Schritt müssen keine weiteren Sperren aktiviert werden. Für die Isolationsstufen Lesestabilität und Cursorstabilität müssen Sperren im zweiten Schritt aktiviert werden. Um den gemeinsamen Zugriff zu maximieren, werden im ersten Schritt keine Sperren aktiviert und alle Vergleichselemente erneut angewandt, so dass nur den Auswahlbedingungen entsprechende Zeilen zurückgegeben werden.

Zugehörige Konzepte:
- „Sperren und Steuerung des gemeinsamen Zugriffs" auf Seite 53
- „Sperrenattribute" auf Seite 55
- „Sperren und Leistung" auf Seite 57
- „Richtlinien für Sperren" auf Seite 62
- „Sperren und Arten der Anwendungsverarbeitung" auf Seite 80
- „Indextypen und Sperren des nächsten Schlüssels" auf Seite 82

Zugehörige Tasks:
- „Korrigieren von Problemen der Sperreneskalation" auf Seite 64

Zugehörige Referenzen:
- „Sperrentypenkompatibilität" auf Seite 69
- „Sperrmodi und Zugriffspfade für Standardtabellen" auf Seite 70
- „Sperrmodi für Tabellen- und Satz-ID-Indexsuchen für MDC-Tabellen" auf Seite 73
- „Sperren für Blockindexsuchen für MDC-Tabellen" auf Seite 77

Indextypen und Sperren des nächsten Schlüssels

Wenn Transaktionen Änderungen an Indizes des Typs 1 verursachen, treten Sperren der jeweils nächsten Schlüssel (Next-Key Locking) auf. Für Indizes des Typs 2 findet das Sperren nächster Schlüssel nur in minimalem Ausmaß statt.

- Sperren der nächsten Schlüssel für Indizes des Typs 2:

  Das Sperren eines nächsten Schlüssels findet statt, wenn ein Schlüssel in einen Index eingefügt wird.

  Während der Einfügung eines Schlüssels in einen Index, wird die Zeile, die dem Schlüssel entspricht, der auf den neuen Schlüssel im Index als Nächstes folgt, nur gesperrt, wenn diese Zeile momentan durch eine RR-Indexsuche gesperrt ist. Der Sperrmodus, der für die Sperre des nächsten Schlüssels verwendet wird, ist NW. Diese Sperre des nächsten Schlüssels wird wieder freigegeben, bevor die eigentliche Einfügung des Schlüssels erfolgt. Die Einfügung des Schlüssels erfolgt, wenn eine Zeile in eine Tabelle eingefügt wird.

  Wenn Aktualisierungen an einer Zeile zu einer Änderung am Wert des Indexschlüssels für diese Zeile führen, erfolgt die Einfügung des Schlüssels außerdem deshalb, weil der ursprüngliche Schlüsselwert als gelöscht markiert wird und der neue Schlüsselwert in den Index eingefügt wird. Bei Aktualisierungen, die nur die INCLUDE-Spalten eines Index betreffen, kann der Schlüssel an seinem Platz aktualisiert werden, und es findet keine Sperrung des nächsten Schlüssels statt.

  Bei RR-Suchen wird die Zeile, die dem Schlüssel entspricht, der auf das Ende des Suchbereichs folgt, im S-Modus gesperrt. Wenn auf das Ende des Suchbereichs keine Schlüssel folgen, wird eine Tabellenende-Sperre aktiviert, um das Ende des Index zu sperren. Wenn der Schlüssel, der auf das Ende des Suchbereichs folgt, als gelöscht markiert ist, fährt die Suche mit dem Sperren der entsprechenden Zeilen solange fort, bis ein Schlüssel angetroffen wird, der nicht als gelöscht markiert ist, wenn die Suche die entsprechende Zeile für diesen Schlüssel sperrt, oder solange bis das Ende des Index gesperrt ist.
- Sperren der nächsten Schlüssel für Indizes des Typs 1:

  Sperren der nächsten Schlüssel werden bei Lösch- und Einfügeoperationen an Indizes und bei Indexsuchen aktiviert. Wenn eine Zeile in einer Tabelle aktualisiert, aus ihr gelöscht oder in sie eingefügt wird, wird für diese Zeile eine X-Sperre aktiviert. Bei Einfügungen kann der Sperrmodus auf eine W-Sperre herabgesetzt werden.

Wenn der Schlüssel aus dem Tabellenindex gelöscht oder in ihn eingefügt wird, wird die Tabellenzeile gesperrt, die dem Schlüssel entspricht, der auf den gelöschten oder eingefügten Schlüssel im Index folgt. Bei Aktualisierungen, die den Wert des Schlüssels betreffen, wird der ursprüngliche Schlüsselwert zuerst gelöscht und der neue Wert eingefügt, so dass zwei Sperren für nächste Schlüssel aktiviert werden. Die Dauer dieser Sperren ist wie folgt festgelegt:

- Beim Löschen eines Indexschlüssels ist der Sperrmodus für den nächsten Schlüssel X und die Sperre bleibt bis zum Festschreiben (COMMIT) aktiv.
- Beim Einfügen eines Indexschlüssels ist der Sperrmodus für den nächsten Schlüssel NW. Diese Sperre wird nur aktiviert, wenn es eine Konkurrenzsituation für die Sperre gibt. In diesem Fall wird die Sperre freigegeben, bevor der Schlüssel tatsächlich in den Index eingefügt wird.
- Bei RR-Suchen wird die Tabellenzeile, die dem Schlüssel direkt nach dem Ende des Indexsuchbereichs entspricht, im Modus S gesperrt und die Sperre bis zum Festschreiben (COMMIT) beibehalten.
- Bei CS/RS-Suchen wird die Zeile, die dem Schlüssel direkt nach dem Ende des Indexsuchbereichs entspricht, im Modus NS gesperrt, wenn eine Konkurrenzsituation für die Sperre vorliegt. Diese Sperre wird freigegeben, wenn das Ende des Suchbereichs einmal geprüft ist.

Das Sperren nächster Schlüssel für Indizes des Typs 1 bei Einfügungen und Löschungen von Schlüsseln kann zu gegenseitigen Sperren führen. Das folgende Beispiel zeigt, wie sich zwei Transaktionen gegenseitig sperren könnten. Bei Indizes des Typs 2 treten solche gegenseitigen Sperren nicht auf.

Betrachten Sie das folgende Beispiel eines Index, der sechs Zeilen mit den folgenden Werten enthält: `1 5 6 7 8 12`.

1. Transaktion 1 löscht die Zeile mit dem Schlüsselwert 8. Die Zeile mit dem Wert 8 wird im Modus X gesperrt. Wenn der entsprechende Schlüssel aus dem Index gelöscht wird, wird die Zeile mit dem Wert 12 im Modus X gesperrt.
2. Transaktion 2 löscht die Zeile mit dem Schlüsselwert 5. Die Zeile mit dem Wert 5 wird im Modus X gesperrt. Wenn der entsprechende Schlüssel aus dem Index gelöscht wird, wird die Zeile mit dem Wert 6 im Modus X gesperrt.
3. Transaktion 1 fügt eine Zeile mit dem Schlüsselwert 4 ein. Diese Zeile wird im Modus W gesperrt. Wenn versucht wird, den neuen Schlüssel in den Index einzufügen, wird die Zeile mit dem Wert 6 im Modus NW gesperrt. Dieser Sperrversuch muss auf die X-Sperre warten, die von Transaktion 2 für diese Zeile aktiviert wurde.
4. Transaktion 2 fügt eine Zeile mit dem Schlüsselwert 9 ein. Diese Zeile wird im Modus W gesperrt. Wenn versucht wird, den neuen Schlüssel in den Index einzufügen, wird die Zeile mit dem Schlüsselwert 12 im Modus NW gesperrt. Dieser Sperrversuch muss auf die X-Sperre warten, die von Transaktion 1 für diese Zeile aktiviert wurde.

Bei der Verwendung von Indizes des Typs 1 führt ein solches Szenario zu einer gegenseitigen Sperre und eine der Transaktionen wird rückgängig gemacht (ROLLBACK).

Zugehörige Konzepte:

- „Vor- und Nachteile von Indizes" auf Seite 288
- „Indexleistung - Tipps" auf Seite 293
- „Indexstruktur" auf Seite 27
- „Indexreorganisation" auf Seite 298
- „Online-Indexdefragmentierung" auf Seite 300
- „Indexbereinigung und Indexpflege" auf Seite 296

Optimierungsfaktoren

Dieser Abschnitt beschreibt die Faktoren, die bei der Angabe der Optimierungsklasse für Abfragen zu berücksichtigen sind.

Richtlinien für Optimierungsklassen

Wenn Sie eine SQL-Abfrage kompilieren, können Sie eine Optimierungsklasse angeben, die bestimmt, wie das Optimierungsprogramm den effizientesten Zugriffsplan für diese Abfrage auswählt. Obwohl Sie einerseits die Optimierungstechniken einzeln angeben können, um die Laufleistung für die Abfrage zu verbessern, werden andererseits um so mehr Zeit und Systemressourcen für die Abfragekompilierung benötigt, je mehr Optimierungstechniken Sie angeben.

Anmerkung: In einer Abfrage für zusammengeschlossene Datenbanken gilt die Optimierungsklasse nicht für das ferne Optimierungsprogramm.

Die Einstellung der Optimierungsklasse kann, insbesondere zu folgenden Zwecken, einige der Vorteile bringen, die durch die explizite Angabe von Optimierungstechniken erzielt werden:
- Verwalten sehr kleiner Datenbanken oder sehr einfacher dynamischer Abfragen
- Kompilieren mit beschränkten Speicherkapazitäten auf dem Datenbankserver
- Verkürzen der Abfragekompilierzeit, zum Beispiel für PREPARE

Die meisten Anweisungen lassen sich in angemessener Weise bei einem sinnvollen Einsatz von Ressourcen unter Verwendung der Standardoptimierungsklasse 5 optimieren. Bei einer bestimmten Optimierungsklasse werden die Kompilierzeit und der Bedarf an Ressourcen in erster Linie durch die Komplexität der Abfrage, insbesondere durch die Anzahl der Verknüpfungen und Unterabfragen, beeinflusst. Allerdings werden die Kompilierzeit und der Ressourcenbedarf auch vom Grad der durchgeführten Optimierung beeinflusst.

Die Abfrageoptimierungsklassen 1, 2, 3, 5 und 7 sind alle für allgemeine Zwecke geeignet. Ziehen Sie die Klasse 0 nur in Betracht, wenn Sie die Kompilierzeit weiter verringern müssen und wissen, das die SQL-Anweisungen extrem einfach sind.

Tipp: Zur Analyse von Abfragen mit langer Laufzeit können Sie die Abfragen mit db2batch ausführen, um herauszufinden, wie viel Zeit auf die Kompilierung und wie viel Zeit auf die Ausführung verwendet wird. Wenn die Kompilierung mehr Zeit erfordert, verringern Sie die Optimierungsklasse. Wenn die Ausführung mehr Zeit erfordert, ziehen Sie eine höhere Optimierungsklasse in Betracht.

Beachten Sie bei der Auswahl einer Optimierungsklasse die folgenden allgemeinen Richtlinien:
- Beginnen Sie mit der Standardoptimierungsklasse für Abfragen: Klasse 5.
- Zur Verwendung einer anderen als die Standardklasse, versuchen Sie zunächst die Klasse 1, 2 oder 3. Die Klassen 0, 1 und 2 arbeiten mit dem Algorithmus der schnellen Verknüpfungsaufzählung (Greedy Join Enumeration).
- Verwenden Sie die Optimierungsklasse 1 oder 2, wenn Sie viele Tabellen mit vielen Verknüpfungsvergleichselementen für dieselbe Spalte haben und die Dauer der Kompilierung von Bedeutung ist.
- Verwenden Sie eine niedrige Optimierungsklasse (0 oder 1) für Abfragen, die sehr kurze Laufzeiten von unter einer Sekunde haben. Solche Abfragen sind häufig durch folgende Merkmale gekennzeichnet:
  - Zugriff auf eine oder nur einige wenige Tabellen

  - Abruf nur einer oder einiger weniger Zeilen
  - Verwendung vollständig qualifizierter eindeutiger Indizes

  OLTP-Transaktionen (Online-Transaktionsverarbeitung) sind gute Beispiele für diese Art von SQL.
- Verwenden Sie eine höhere Optimierungsklasse (3, 5 oder 7) für Abfragen mit längerer Laufzeit, die mehr als 30 Sekunden benötigen.
- Klasse 3 und höhere Klassen arbeiten mit dem Algorithmus zur dynamisch programmierten Verknüpfungsaufzählung (Dynamic Programming Join Enumeration). Dieser Algorithmus berücksichtigt wesentlich mehr alternative Pläne und kann, insbesondere bei steigender Tabellenzahl, deutlich mehr Kompilierungszeit als die Klassen 0, 1 und 2 erfordern.
- Verwenden Sie die Optimierungsklasse 9 nur, wenn Sie spezielle, über das normale Maß hinausgehende Optimierungsanforderungen für eine Abfrage haben.

Für komplexe Abfragen können andere Grade an Optimierung erforderlich sein, um den besten Zugriffsplan auszuwählen. Ziehen Sie höhere Optimierungsklassen für Abfragen mit den folgenden Merkmalen in Betracht:
- Zugriff auf große Tabellen
- Eine große Anzahl von Vergleichselementen
- Zahlreiche Unterabfragen
- Zahlreiche Verknüpfungen
- Zahlreiche Mengenoperatoren wie UNION und INTERSECT
- Zahlreiche die Vergleichselemente erfüllende Zeilen
- Operationen GROUP BY und HAVING
- Verschachtelte Tabellenausdrücke
- Eine große Anzahl von Sichten

Abfragen zur Entscheidungshilfe oder Abfragen für Monatsberichte aus vollständig normalisierten Datenbanken sind gute Beispiele für komplexe Abfragen, für die zumindest die Standardoptimierungsklasse verwendet werden sollte. Verwenden Sie höhere Optimierungsklassen für SQL, das von einem Abfragegenerator erstellt wurde. Viele Abfragegeneratoren erstellen ineffizientes SQL. Ineffizient geschriebene Abfragen, einschließlich der von einem Abfragegenerator erstellten, können eine zusätzliche Optimierung erforderlich machen, um einen guten Zugriffsplan auszuwählen. Durch Verwendung der Abfrageoptimierungsklasse 2 oder einer höheren können solche SQL-Abfragen verbessert werden.

Zugehörige Konzepte:
- „Konfigurationsparameter mit Einfluss auf die Abfrageoptimierung" auf Seite 161
- „Durchführen von Vergleichstests" auf Seite 357
- „Optimierungsstrategien für partitionsinterne Parallelität" auf Seite 203
- „Optimierungsstrategien für MDC-Tabellen" auf Seite 206

Zugehörige Tasks:
- „Einstellen der Optimierungsklasse" auf Seite 89

Zugehörige Referenzen:
- „Optimierungsklassen" auf Seite 86

Optimierungsklassen

Bei der Kompilierung einer SQL-Abfrage können Sie eine der folgenden Optimierungsklassen angeben:

0 - Diese Klasse weist das Optimierungsprogramm an, nur eine Minimaloptimierung zur Generierung eines Zugriffsplans durchzuführen. Diese Optimierungsklasse ist durch folgende Merkmale gekennzeichnet:
- Statistiken über ungleichmäßige Verteilungen von Datenwerten werden vom Optimierungsprogramm nicht berücksichtigt.
- Nur Grundregeln für das Umschreiben der Abfragen werden angewandt.
- Schnelle Verknüpfungsaufzählung findet statt.
- Nur die Zugriffsmethoden durch Verknüpfungen über Verschachtelungsschleife und Indexsuchen sind möglich.
- Ein Vorablesezugriff über Listen und logisches Verknüpfen von Indizes über AND (Index ANDing) werden in den generierten Zugriffsmethoden nicht verwendet.
- Die Strategie der Sternverknüpfung (Star Join) wird nicht berücksichtigt.

Diese Klasse sollte nur unter Umständen verwendet werden, unter denen der Systemaufwand zur Kompilierung der Abfrage so gering wie möglich gehalten werden muss. Die Abfrageoptimierungsklasse 0 eignet sich für eine Anwendung, die insgesamt aus sehr einfachen dynamischen SQL-Anweisungen besteht, die auf zweckmäßig indexierte Tabellen zugreifen.

1 - Diese Optimierungsklasse ist durch folgende Merkmale gekennzeichnet:
- Statistiken über ungleichmäßige Verteilungen von Datenwerten werden vom Optimierungsprogramm nicht berücksichtigt.
- Nur eine Teilmenge der Regeln für das Umschreiben von Abfragen wird angewandt.
- Schnelle Verknüpfungsaufzählung findet statt.
- Ein Vorablesezugriff über Listen und logisches Verknüpfen von Indizes über AND (Index ANDing) werden in den generierten Zugriffsmethoden nicht verwendet, obwohl das logische Verknüpfen von Indizes über AND für die Bearbeitung einfacher Gleichheitsverknüpfungen (Semijoins) in Sternverknüpfungen verwendet wird.

Die Optimierungsklasse 1 ist der Klasse 0 ähnlich, abgesehen davon, dass Mischverknüpfungen und Tabellensuchen ebenfalls verfügbar sind.

2 - Diese Klasse weist das Optimierungsprogramm an, einen Optimierungsgrad zu verwenden, der den der Klasse 1 deutlich übertrifft, und gleichzeitig den Kompilierungsaufwand für komplexe Abfragen wesentlich geringer als bei den Klassen ab 3 aufwärts zu halten. Diese Optimierungsklasse ist durch folgende Merkmale gekennzeichnet:
- Alle verfügbaren Statistiken, einschließlich der Statistiken zur Häufigkeit und zu Quantilen ungleichmäßiger Verteilungen, werden verwendet.
- Alle Regeln für das Umschreiben von Abfragen, einschließlich der Weiterleitung von Abfragen an gespeicherte Abfragetabellen, werden berücksichtigt, außer den Regeln, die sehr rechenintensiv sind und nur in seltenen Fällen zur Anwendung kommen.
- Schnelle Verknüpfungsaufzählung wird verwendet.
- Viele verschiedene Zugriffsmethoden werden berücksichtigt, einschließlich des Vorablesezugriffs über Listen und der Weiterleitung an gespeicherte Abfragetabellen.
- Die Strategie der Sternverknüpfung (Star Join) wird gegebenenfalls berücksichtigt.

Die Optimierungsklasse 2 ist der Klasse 5 ähnlich, sie verwendet jedoch schnelle Verknüpfungsaufzählung und nicht dynamische programmierte Verknüpfungsaufzählung. Diese Klasse hat den höchsten Optimierungsgrad aller Klassen, die mit dem Algorithmus für schnelle Verknüpfungsaufzählung arbeiten, der für komplexe Abfragen weniger Alternativen berücksichtigt und dadurch einen geringeren Kompilierungsaufwand erfordert als die Klassen ab 3 aufwärts. Klasse 2 empfiehlt sich für sehr komplexe Abfragen in einer Umgebung zur Entscheidungshilfe oder mit analytischer Onlineverarbeitung (OLAP). In solchen Umgebungen werden spezifische Abfragen nur selten exakt wiederholt, so dass ein Zugriffsplan mit hoher Wahrscheinlichkeit nicht bis zur nächsten Ausführung der Abfrage im Cache erhalten bleibt.

3 - Diese Klasse fordert einen moderaten Optimierungsrad an. Diese Klasse kommt den Merkmalen der Abfrageoptimierung von DB2 für MVS/ESA, OS/390 oder z/OS am nächsten. Diese Optimierungsklasse ist durch folgende Merkmale gekennzeichnet:

- Statistiken über ungleichmäßige Verteilungen von Datenwerten, die die Häufigkeit von Werten erfassen, werden verwendet, wenn sie verfügbar sind.
- Die meisten Regeln zum Umschreiben von Abfragen, einschließlich der Umsetzungen von Unterabfragen in Verknüpfungen, werden angewandt.
- Dynamisch programmierte Verknüpfungsaufzählung (Dynamic Programming Join Enumeration) wie folgt:
  - Eingeschränkte Verwendung zusammengesetzter innerer Tabellen
  - Eingeschränkte Verwendung kartesischer Produkte für Sternschemata, für die Suchtabellen erforderlich sind
- Viele verschiedene Zugriffsmethoden werden berücksichtigt, einschließlich des Vorablesezugriffs über Listen, logisches Verknüpfen von Indizes über AND (Index ANDing) und Sternverknüpfungen (Star Joins).

Diese Klasse eignet sich für eine große Bandbreite von Anwendungen. Diese Klasse verbessert Zugriffspläne für Abfragen mit vier und mehr Verknüpfungen. Es ist jedoch möglich, dass der vom Optimierungsprogramm gewählte Plan nicht so gut ist wie einer, der mit der Standardoptimierungsklasse gewählt würde.

5 - Diese Klasse weist das Optimierungsprogramm an, einen bedeutenden Grad an Optimierung bei der Generierung eines Zugriffsplans durchzuführen. Diese Optimierungsklasse ist durch folgende Merkmale gekennzeichnet:

- Alle verfügbaren Statistiken, einschließlich der Verteilungsstatistiken zur Häufigkeit und zu Quantilen, werden verwendet.
- Alle Regeln für das Umschreiben von Abfragen, einschließlich der Weiterleitung von Abfragen an gespeicherte Abfragetabellen, werden berücksichtigt, außer den Regeln, die sehr rechenintensiv sind und nur in seltenen Fällen zur Anwendung kommen.
- Dynamisch programmierte Verknüpfungsaufzählung (Dynamic Programming Join Enumeration) wie folgt:
  - Eingeschränkte Verwendung zusammengesetzter innerer Tabellen
  - Eingeschränkte Verwendung kartesischer Produkte für Sternschemata, für die Suchtabellen erforderlich sind

- Viele verschiedene Zugriffsmethoden werden berücksichtigt, einschließlich des Vorablesezugriffs über Listen, des logischen Verknüpfens von Indizes über AND und der Weiterleitung an gespeicherte Abfragetabellen.

Stellt das Optimierungsprogramm fest, dass die zusätzlichen Ressourcen und die Verarbeitungszeit für komplexe dynamische SQL-Abfragen nicht gewährleistet sind, wird die Optimierung reduziert. Das Ausmaß oder der Umfang der Reduzierung hängt von der Maschinengröße und der Anzahl der Vergleichselemente ab.

Reduziert das Abfrageoptimierungsprogramm den Grad an Abfrageoptimierung, wendet es weiterhin alle Regeln für das Umschreiben von Abfragen an, die normalerweise angewandt würden. Es verwendet jedoch die schnelle Verknüpfungszählung und reduziert die Anzahl der Zugriffsplankombinationen, die in Erwägung gezogen werden.

Die Abfrageoptimierungsklasse 5 ist hervorragend für eine gemischte Umgebung geeignet, in der sowohl Transaktionen als auch komplexe Abfragen ausgeführt werden. Diese Optimierungsklasse wurde zur Verwendung der wertvollsten Abfragetransformationen und anderer Optimierungstechniken für Abfragen in einer effizienten Weise entwickelt.

7 - Diese Klasse weist das Optimierungsprogramm an, einen bedeutenden Grad an Optimierung bei der Generierung eines Zugriffsplans durchzuführen. Sie entspricht der Abfrageoptimierungsklasse 5, jedoch reduziert sie nicht den Grad der Abfrageoptimierung für komplexe dynamische SQL-Abfragen.

9 - Diese Klasse weist das Optimierungsprogramm an, alle verfügbaren Optimierungstechniken anzuwenden. Dazu gehören:

- Alle verfügbaren Statistiken
- Alle Regeln für das Umschreiben von Abfragen
- Alle Möglichkeiten für Verknüpfungsaufzählungen, einschließlich kartesischer Produkte und uneingeschränkter zusammengesetzter innerer Tabellen
- Alle Zugriffsmethoden

Diese Klasse kann die Anzahl der möglichen Zugriffspläne, die vom Optimierungsprogramm ausgewertet werden, erheblich vergrößern. Diese Klasse kann verwendet werden, um festzustellen, ob eine umfassendere Optimierung zur Generierung eines besseren Zugriffsplans für sehr komplexe und zeitintensive Abfragen auf große Tabellen führen würde. Überprüfen Sie anhand von EXPLAIN-Daten und Leistungswerten, ob ein besserer Plan gefunden wurde.

Zugehörige Konzepte:

- „Richtlinien für Optimierungsklassen" auf Seite 84
- „Optimierungsstrategien für partitionsinterne Parallelität" auf Seite 203
- „Generierung von fernem SQL und globale Optimierung in zusammengeschlossenen Datenbanken" auf Seite 217
- „Optimierungsstrategien für MDC-Tabellen" auf Seite 206

Zugehörige Tasks:

- „Einstellen der Optimierungsklasse" auf Seite 89

Einstellen der Optimierungsklasse

Berücksichtigen Sie bei der Angabe einer Optimierungsstufe, ob eine Abfrage statisches oder dynamisches SQL verwendet und ob das gleiche dynamische SQL wiederholt ausgeführt wird. Für statisches SQL tritt der Aufwand an Kompilierzeit und Ressourcen nur einmal auf, und der ausgewählte Zugriffsplan kann mehrfach verwendet werden. Im Allgemeinen gilt, dass für statisches SQL stets die Standardoptimierungsklasse verwendet werden sollte. Da dynamische Anweisungen bei der Ausführung gebunden und ausgeführt werden, müssen Sie überlegen, ob der Aufwand für eine zusätzliche Optimierung der dynamischen Anweisungen die allgemeine Leistung verbessert. Wenn dieselbe dynamische SQL-Anweisung jedoch wiederholt ausgeführt wird, wird der ausgewählte Zugriffsplan im Cache zwischengespeichert. Solche Anweisungen können dieselben Optimierungsstufen wie statische SQL-Anweisungen verwenden.

Wenn Sie annehmen, dass für eine Abfrage eine weitere Optimierung von Vorteil wäre, Sie aber nicht sicher sind oder Bedenken hinsichtlich der Kompilierzeit oder des Ressourcenbedarfs haben, können Sie einige Vergleichstests (Benchmarktests) durchführen.

Vorgehensweise:

Führen Sie folgende Schritte aus, um eine Abfrageoptimierungsklasse anzugeben:

1. Analysieren Sie die Leistungsfaktoren entweder informell oder wie folgt mit formalen Tests:
   - Für **dynamische** SQL-Anweisungen sollten Tests die durchschnittliche Laufzeit für die Anweisung vergleichen. Schätzen Sie eine durchschnittliche Laufzeit mit Hilfe folgender Formel ab:

     ```
     Kompilierzeit + Summe der Ausführungszeiten aller Iterationen
     -------------------------------------------------------------
                    Anzahl der Iterationen
     ```

     In dieser Formel ist die `Anzahl der Iterationen` die Häufigkeit, mit der die SQL-Anweisung Ihrer Schätzung nach jedes Mal, wenn sie kompiliert wird, ausgeführt werden könnte.

     Anmerkung: Nach der Erstkompilierung werden dynamische SQL-Anweisungen erneut kompiliert, wenn eine Änderung an der Umgebung dies erfordert. Wenn sich die Umgebung nicht ändert, nachdem eine SQL-Anweisung im Cache zwischengespeichert wurde, braucht sie nicht erneut kompiliert zu werden, da nachfolgende PREPARE-Anweisungen die zwischengespeicherte Anweisung wiederverwenden.
   - Vergleichen Sie für **statische** SQL-Anweisungen die Laufzeiten der Anweisungen.

     Obwohl es vielleicht auch interessant wäre, die Kompilierzeit statischer SQL-Anweisungen zu kennen, ist die Gesamtzeit aus Kompilier- und Laufzeit der Anweisung in einem sinnvollen Kontext nur wenig aussagekräftig. Beim Vergleich der Gesamtzeit wird die Tatsache außer Acht gelassen, dass eine statische SQL-Anweisung nach jedem Binden viele Male ausgeführt werden kann und dass sie in der Regel nicht während der Laufzeit gebunden wird.
2. Geben Sie die Optimierungsklasse wie folgt an:
   - **Dynamische** SQL-Anweisungen verwenden die Optimierungsklasse, die im Sonderregister CURRENT QUERY OPTIMIZATION angegeben ist, das mit

der SQL-Anweisung SET definiert wird. Beispielsweise setzt die folgende Anweisung die Optimierungsklasse auf 1:

```
SET CURRENT QUERY OPTIMIZATION = 1
```

Um sicherzustellen, dass eine dynamische SQL-Anweisung immer dieselbe Optimierungsklasse verwendet, kann diese Anweisung SET in das Anwendungsprogramm mit aufgenommen werden.

Wenn das Register CURRENT QUERY OPTIMIZATION nicht definiert ist, werden die dynamischen Anweisungen mit der Standardoptimierungsklasse für Abfragen gebunden. Der Standardwert für dynamisches und statisches SQL wird durch den Wert des Datenbankkonfigurationsparameters *dft_queryopt* festgelegt. Der Standardwert für diesen Parameter ist Klasse 5. Die Standardwerte für die Bindeoption und das Sonderregister werden ebenfalls aus dem Datenbankkonfigurationsparameter *dft_queryopt* gelesen.

- **Statische** SQL-Anweisungen verwenden die in den Befehlen PREP und BIND angegebene Optimierungsklasse. In der Spalte QUERYOPT in der Katalogtabelle SYSCAT.PACKAGES wird die zum Binden des Pakets verwendete Optimierungsklasse aufgezeichnet. Wenn das Paket erneut implizit oder mit dem Befehl REBIND PACKAGE gebunden wird, wird dieselbe Optimierungsklasse für die statischen SQL-Anweisungen verwendet. Verwenden Sie zum Ändern der Optimierungsklasse für solche statischen SQL-Anweisungen den Befehl BIND. Wenn Sie keine Optimierungsklasse angeben, verwendet DB2 die vom Datenbankkonfigurationsparameter *dft_queryopt* angegebene Standardoptimierungsklasse.

Zugehörige Konzepte:

- „Richtlinien für Optimierungsklassen" auf Seite 84

Zugehörige Referenzen:

- „Optimierungsklassen" auf Seite 86

Optimieren von Anwendungen

Dieser Abschnitt enthält Richtlinien zur Optimierung von Abfragen, die von Anwendungen ausgeführt werden.

Richtlinien zur Begrenzung von SELECT-Anweisungen

Das Optimierungsprogramm geht von der Annahme aus, dass eine Anwendung sämtliche Zeilen abrufen muss, die in der SELECT-Anweisung angegeben sind. Diese Annahme ist in OLTP-Umgebungen und bei Stapelbetrieb in der Regel zutreffend. Bei reinen Such- bzw. Anzeigeanwendungen („Browse") hingegen werden durch die Abfragen häufig potenziell sehr umfangreiche Antwortmengen definiert, von denen jedoch normalerweise nur so viele Zeilen abgerufen werden, wie auf eine Bildschirmanzeige passen.

Zur Verbesserung der Leistung solcher Anwendungen können Sie die SELECT-Anweisung auf folgende Weisen modifizieren:

- Verwenden Sie die Klausel FOR UPDATE, um die Spalten anzugeben, die von einer nachfolgenden positionierten Anweisung UPDATE aktualisiert werden könnten.
- Verwenden Sie die Klausel FOR READ/FETCH ONLY, um die angegebenen Spalten im Lesezugriff zurückzuliefern.

- Verwenden Sie die Klausel OPTIMIZE FOR *n* ROWS, um dem Abrufen der ersten *n* Zeilen der Gesamtergebnismenge Priorität geben.
- Verwenden Sie die Klausel FETCH FIRST *n* ROWS ONLY, um nur eine angegebene Anzahl von Zeilen abzurufen.
- Verwenden Sie die Anweisung DECLARE CURSOR WITH HOLD, um jeweils nur eine Zeile pro Mal abzurufen.

Anmerkung: Die Verwendung der Klausel FOR UPDATE, FETCH FIRST *n* ROWS ONLY oder OPTIMIZE FOR *n* ROWS bzw. die Deklaration des Cursors mit SCROLL hat Auswirkung auf die Zeilenblockung.

In den folgenden Abschnitten werden die Leistungsvorteile der einzelnen Methoden erläutert.

Klausel FOR UPDATE

Die Klausel FOR UPDATE begrenzt die Ergebnismenge ein, indem sie nur die Spalten berücksichtigt, die durch eine nachfolgende positionierte UPDATE-Anweisung aktualisiert werden können. Bei der Angabe der Klausel FOR UPDATE ohne Spaltennamen werden alle Spalten, die aktualisiert werden können, in der Tabelle oder Sicht mit eingeschlossen. Bei Angabe von Spaltennamen muss jeder Name unqualifiziert angegeben werden und eine Spalte der Tabelle oder Sicht bezeichnen.

Unter folgenden Umständen können Sie die Klausel FOR UPDATE nicht verwenden:

- Wenn der Cursor, der der der SELECT-Anweisung zugeordnet ist, nicht gelöscht werden kann.
- Wenn mindestens eine der durch SELECT ausgewählten Spalten eine Spalte ist, die nicht in einer Katalogtabelle aktualisiert werden kann und in der Klausel FOR UPDATE nicht ausgeschlossen wurde.

Verwenden Sie in CLI-Anwendungen zu den gleichen Zwecken das DB2® CLI-Verbindungsattribut SQL_ATTR_ACCESS_MODE.

Klausel FOR READ oder FETCH ONLY

Die Klausel FOR READ ONLY oder FOR FETCH ONLY stellt sicher, dass Ergebnisse nur im Lesezugriff zugeliefert werden. Da die Ergebnistabelle aus einer SELECT-Operation für eine Sicht, die als schreibgeschützt definiert ist, ebenfalls schreibgeschützt ist, ist diese Klausel zulässig, jedoch hat sie keinen Effekt.

Bei Tabellen, für die Aktualisierungen und Löschungen zulässig sind, kann die Angabe der Klausel FOR READ ONLY die Leistung von Abrufoperationen (FETCH) verbessern, wenn der Datenbankmanager Blöcke von Daten abrufen kann, statt exklusive Sperren zu aktivieren. Verwenden Sie die Klausel FOR READ ONLY nicht für Abfragen, in denen positionierte UPDATE- oder DELETE-Anweisungen verwendet werden.

Das DB2 CLI-Verbindungsattribut SQL_ATTR_ACCESS_MODE kann zu denselben Zwecken in CLI-Anwendungen verwendet werden.

Klausel OPTIMIZE FOR n ROWS

Mit der Klausel OPTIMIZE FOR wird die Absicht deklariert, nur eine Teilmenge des Ergebnisses abzurufen oder dem Abruf nur der ersten wenigen Zeilen Priorität zu geben. Das Optimierungsprogramm kann in diesem Fall Zugriffspläne bevorzugen, die die Antwortzeiten für den Abruf der ersten wenigen Zeilen minimieren. Darüber hinaus wird die Anzahl der Zeilen, die als ein Block an den Client gesendet werden, durch den Wert „n" der Klausel OPTIMIZE FOR begrenzt. Daher beeinflusst die Klausel OPTIMIZE FOR sowohl die Art und Weise, wie der Server die den Bedingungen entsprechenden Zeilen aus der Datenbank abruft, als auch die Art und Weise, wie er diese Zeilen an den Client zurückgibt.

Nehmen Sie zum Beispiel an, Sie führen regelmäßig eine Abfrage nach den Mitarbeitern mit den höchsten Gehältern auf die Tabelle EMPLOYEE aus.

```
SELECT LASTNAME,FIRSTNAME,EMPNO,SALARY
FROM EMPLOYEE
ORDER BY SALARY DESC
```

Sie haben einen absteigenden Index für die Spalte SALARY definiert. Da jedoch die Mitarbeiter nach der Personalnummer (Spalte EMPNO) geordnet sind, weist der Index für die Spalte SALARY wahrscheinlich eine geringe Clusterbildung auf. Zur Vermeidung zahlreicher synchroner E/A-Operationen wählt das Optimierungsprogramm wahrscheinlich die Methode des Vorablesezugriffs über Listen, für die eine Sortierung der Satz-IDs (RIDs) aller qualifizierten Zeilen erforderlich ist. Diese Sortierung führt zu einer Verzögerung, bevor die ersten Ergebniszeilen an die Anwendung zurückgegeben werden. Zur Vermeidung dieser Verzögerung können Sie der Anweisung die Klausel OPTIMIZE FOR wie folgt hinzufügen:

```
SELECT LASTNAME,FIRSTNAME,EMPNO,SALARY
FROM EMPLOYEE
ORDER BY SALARY DESC
OPTIMIZE FOR 20 ROWS
```

In diesem Fall geht das Optimierungsprogramm wahrscheinlich so vor, dass es den Index SALARY direkt verwendet, weil nur die 20 Mitarbeiter mit den höchsten Gehältern abgerufen werden. Unabhängig davon, wie viele Zeilen geblockt werden könnten, wird nun alle zwanzig Zeilen ein Zeilenblock an den Client zurückgegeben. Mit der Klausel OPTIMIZE FOR bevorzugt das Optimierungsprogramm Zugriffspläne, die Massenoperationen oder die Unterbrechung des Zeilenflusses, zum Beispiel durch Sortierungen, vermeiden. Am wahrscheinlichsten wird ein Zugriffspfad durch die Klausel OPTIMIZE FOR 1 ROW beeinflusst. Die Verwendung dieser Klausel kann folgende Auswirkungen haben:

- Tabellenverknüpfungssequenzen mit zusammengesetzten inneren Tabellen treten mit geringerer Wahrscheinlichkeit auf, da für sie eine temporäre Tabelle erforderlich ist.
- Die Verknüpfungsmethode könnte sich ändern. Eine Verknüpfung über Verschachtelungsschleife (Nested Loop Join) wird mit größter Wahrscheinlichkeit gewählt, da diese Methode relativ geringen Aufwand verursacht und in der Regel zum Abrufen einiger weniger Zeilen effizienter ist.
- Ein Index, der der Klausel ORDER BY entspricht, wird mit größerer Wahrscheinlichkeit genutzt, weil dadurch die Sortierung für die Klausel ORDER BY entfällt.
- Der Vorablesezugriff über Listen wird mit geringerer Wahrscheinlichkeit verwendet, da diese Zugriffsmethode eine Sortierung erforderlich macht.
- Ein sequenzieller Vorablesezugriff ist weniger wahrscheinlich, da bekannt ist, dass nur eine kleine Anzahl von Zeilen erforderlich ist.

- Bei einer Verknüpfungsabfrage wird wahrscheinlich die Tabelle mit den Spalten in der Klausel ORDER BY als äußere Tabelle gewählt, wenn ein Index für die äußere Tabelle die Reihenfolge bietet, die für die Klausel ORDER BY erforderlich ist.

Obwohl die Klausel OPTIMIZE FOR für alle Optimierungsklassen gültig ist, zeigt sie für Klassen ab Optimierungsklasse 3 die besten Ergebnisse, weil die Klassen unter 3 mit der Methode der schnellen Verknüpfungsaufzählung (Greedy Join Enumeration) arbeiten. Diese Methode führt manchmal zu Zugriffsplänen für Verknüpfungen mehrerer Tabellen, die für ein schnelles Abrufen der ersten Zeilen nicht geeignet sind.

Die Klausel OPTIMIZE FOR bewirkt nicht, dass das Abrufen aller Ergebniszeilen unmöglich wird. Wenn Sie doch alle Ergebniszeilen abrufen, kann die Gesamtdauer wesentlich länger sein, als wenn das Optimierungsprogramm für die gesamte Antwortmenge optimiert hätte.

Wenn eine Paketanwendung das Call Level Interface (DB2 CLI oder ODBC) verwendet, können Sie das Schlüsselwort OPTIMIZEFORNROWS in der Konfigurationsdatei `db2cli.ini` verwenden, um DB2 CLI zu veranlassen, automatisch eine Klausel OPTIMIZE FOR an das Ende jeder Abfrageanweisung anzuhängen.

Beim Auswählen von Daten aus Kurznamen können die Ergebnisse abhängig von der Unterstützung durch die Datenquelle unterschiedlich ausfallen. Wenn die durch den Kurznamen angegebene Datenquelle die Klausel OPTIMIZE FOR unterstützt und das Optimierungsprogramm die gesamte Abfrage an die Datenquelle verschiebt (Pushdown), wird die Klausel im fernen SQL generiert, das an die Datenquelle gesendet wird. Wenn die Datenquelle diese Klausel nicht unterstützt oder das Optimierungsprogramm entscheidet, das der Plan des geringsten Aufwands eine lokale Ausführung ist, wird die Klausel OPTIMIZE FOR lokal in DB2 angewandt. In diesem Fall bevorzugt das DB2-Optimierungsprogramm Zugriffspläne, die die Antwortzeit für das Abrufen der ersten wenigen Zeilen einer Abfrage minimieren, jedoch sind die dem Optimierungsprogramm zur Verfügung stehenden Optionen zur Generierung von Plänen etwas beschränkt und die Leistungsgewinne aus der Klausel OPTIMIZE FOR möglicherweise vernachlässigbar.

Wenn sowohl die Klausel FETCH FIRST als auch die Klausel OPTIMIZE FOR angegeben werden, wirkt sich der niedrigere der beiden Werte auf die Größe des Kommunikationspuffers aus. Die beiden Werte werden hinsichtlich der Optimierung als unabhängig voneinander betrachtet.

Klausel FETCH FIRST n ROWS ONLY

Die Klausel FETCH FIRST *n* ROWS ONLY legt die maximale Anzahl von Zeilen fest, die abgerufen werden können. Die Beschränkung der Ergebnistabelle auf die ersten wenigen Zeilen kann die Leistung erhöhen. Es werden nur *n* Zeilen abgerufen, ungeachtet der Anzahl von Zeilen, die ansonsten in der Ergebnismenge enthalten wären.

Wenn Sie sowohl die Klausel FETCH FIRST als auch die Klausel OPTIMIZE FOR angeben, wirkt sich der niedrigere der beiden Werte auf die Größe des Kommunikationspuffers aus. Hinsichtlich der Optimierung werden die beiden Werte als unabhängig voneinander betrachtet.

Anweisung DECLARE CURSOR WITH HOLD

Wenn Sie einen Cursor mit der Anweisung DECLARE CURSOR deklarieren, die die Klausel WITH HOLD enthält, bleiben alle geöffneten Cursor nach dem Festschreiben der Transaktion geöffnet, und alle Sperren werden freigegeben, mit Ausnahme von Sperren, die die aktuelle Cursorposition geöffneter WITH HOLD-Cursor schützen.

Wenn die Transaktion rückgängig (ROLLBACK) gemacht wird, werden alle geöffneten Cursor geschlossen und alle Sperren und LOB-Querverweise freigegeben.

Durch die Verwendung des DB2-CLI-Verbindungsattributs SQL_ATTR_CURSOR-_HOLD in CLI-Anwendungen können Sie die gleichen Ergebnisse erzielen. Wenn eine Paketanwendung das Call Level Interface (DB2 CLI oder ODBC) verwendet, können Sie das Schlüsselwort CURSORHOLD in der Konfigurationsdatei `db2cli.ini` verwenden, um DB2 CLI zu veranlassen, automatisch die Klausel WITH HOLD für jeden deklarierten Cursor anzunehmen.

Zugehörige Konzepte:

- „Richtlinien zur Abfrageoptimierung" auf Seite 95
- „Effiziente SELECT-Anweisungen" auf Seite 97

Angeben von Zeilenblockung zur Verringerung des Systemaufwands

Zeilenblockung verringert den Systemaufwand des Datenbankmanagers für Cursor durch Abrufen eines *Blocks* von Zeilen in einer einzigen Operation.

Anmerkung: Der Zeilenblock, den Sie angeben, ist eine Anzahl von Seiten im Speicher. Dabei handelt es sich nicht um einen Block einer mehrdimensionalen MDC-Tabelle (MDC - Mehrdimensionales Clustering), der wiederum einem EXTENTSIZE-Speicherbereich auf dem Datenträger zugeordnet ist.

Die Stufen der Zeilenblockung werden in den folgenden Argumenten für die Befehle BIND oder PREP angegeben:

UNAMBIG
: Die Blockung wird für Lesezugriffscursor und für Cursor, die nicht mit „FOR UPDATE OF" definiert sind, verwendet. Mehrdeutige Cursor werden als aktualisierbar betrachtet.

ALL
: Die Blockung wird für Lesezugriffscursor und für Cursor, die nicht mit „FOR UPDATE OF" definiert sind, verwendet. Mehrdeutige Cursor werden wie Lesezugriffscursor behandelt.

NO
: Es wird keine Blockung für Cursor verwendet. Mehrdeutige Cursor werden wie Lesezugriffscursor behandelt.

Voraussetzungen:

Zwei Konfigurationsparameter des Datenbankmanagers müssen entsprechend definiert werden. Beide Werte werden als Anzahl von Seiten im Speicher definiert. Notieren Sie sich die Werte dieser Parameter zur Verwendung in Blockgrößenberechnungen.

- Der Konfigurationsparameter *aslheapsz* des Datenbankmanagers gibt die Größe des Zwischenspeichers der Anwendungsunterstützungsebene an.

- Der Konfigurationsparameter *rqrioblk* des Datenbankmanagers gibt die Größe des Kommunikationspuffers zwischen fernen Anwendungen und ihren Datenbankagenten auf dem Datenbankserver an.

Vorgehensweise:

Gehen Sie wie folgt vor, um die Zeilenblockung zu definieren:

1. Schätzen Sie mit Hilfe der Werte der Konfigurationsparameter *aslheapsz* und *rqrioblk* die Anzahl der Zeilen ab, die für jeden Block zurückgegeben werden. In den beiden folgenden Formeln steht *azl* jeweils für die Ausgabezeilenlänge.
   - Verwenden Sie die folgende Formel für lokale Anwendungen:
     ```
     Zeilen pro Block = aslheapsz * 4096 / azl
     ```
     Die Anzahl von Byte pro Seite beträgt 4 096.
   - Verwenden Sie die folgende Formel für ferne Anwendungen:
     ```
     Zeilen pro Block = rqrioblk / azl
     ```
2. Zur Aktivierung der Zeilenblockung geben Sie ein entsprechendes Argument für die Option BLOCKING in den Befehlen PREP oder BIND an.

   Wenn Sie keine Option BLOCKING angeben, wird standardmäßig der Zeilenblockungstyp UNAMBIG verwendet. Bei Verwendung des Befehlszeilenprozessors und von Call Level Interface ist der Standardtyp der Zeilenblockung ALL.

Anmerkung: Bei Verwendung der Klausel FETCH FIRST *n* ROWS ONLY oder OPTIMIZE FOR *n* ROWS in einer SELECT-Anweisung entspricht die Anzahl der Zeilen pro Block dem kleinsten der folgenden Werte:
- Der mit der oben angegebenen Formel berechnete Wert
- Der Wert von *n* in der Klausel FETCH FIRST
- Der Wert von *n* in der Klausel OPTIMIZE FOR

Zugehörige Referenzen:

- „aslheapsz - Zwischenspeichergröße für Anwendungsunterstützungsebene" auf Seite 421
- „rqrioblk - E/A-Blockgröße für Clients" auf Seite 424

Richtlinien zur Abfrageoptimierung

Befolgen Sie die Richtlinien zur Abfrageoptimierung, um eine optimale Feinabstimmung der SQL-Anweisungen in einem Anwendungsprogramm zu erzielen. Die Richtlinien sollen Ihnen Hilfestellung bei der Minimierung des Bedarfs an Systemressourcen und der Zeit geben, die zur Rückgabe von Ergebnissen aus großen Tabellen und komplexen Abfragen erforderlich ist.

Anmerkung: Die vom Optimierungsprogramm verwendete Optimierungsklasse macht eine Feinabstimmung vielleicht überflüssig, weil der SQL-Compiler den SQL-Code in effizientere Formen umschreiben kann.

Beachten Sie, dass die Auswahl eines Zugriffsplans durch das Optimierungsprogramm auch den Auswirkungen anderer Faktoren, wie zum Beispiel Umgebungsaspekte und Systemkatalogstatistiken, unterliegt. Durch Leistungsvergleichstests können Sie ermitteln, welche Anpassungen möglicherweise die Auswahl eines besseren Zugriffsplan bewirken.

Zugehörige Konzepte:
- „Richtlinien zur Begrenzung von SELECT-Anweisungen" auf Seite 90
- „Effiziente SELECT-Anweisungen" auf Seite 97
- „Richtlinien für Compound-SQL-Anweisungen" auf Seite 99
- „Richtlinien zur Zeichenkonvertierung" auf Seite 101
- „Richtlinien für gespeicherte Prozeduren" auf Seite 102
- „Parallelverarbeitung für Anwendungen" auf Seite 103
- „Richtlinien für die Sortierleistung" auf Seite 278

Zugehörige Tasks:
- „Angeben von Zeilenblockung zur Verringerung des Systemaufwands" auf Seite 94

Stichprobendaten in SQL-Abfragen

Datenbanken wachsen so stark an und die Abfragen für diese Datenbanken werden so komplex, dass es häufig unpraktisch und manchmal auch unnötig ist, auf sämtliche für eine Abfrage relevanten Daten zuzugreifen. In einigen Fällen interessiert sich ein Benutzer für allgemeine Trends oder Muster, bei denen angenäherte Antworten innerhalb gewisser Fehlertoleranzen vollkommen ausreichen. Eine Möglichkeit, solche Abfragen zu beschleunigen, besteht darin, die Abfrage an einer zufälligen Stichprobe der Datenbank auszuführen. DB2® ermöglicht Ihnen, eine effiziente Datenstichprobe in SQL-Abfragen zu erstellen, wodurch sich potenzielle Leistungsverbesserungen für umfangreiche Abfragen um mehrere Größenordnungen ergeben und gleichzeitig ein hoher Grad an Genauigkeit bewahrt werden kann.

Die gängigste Anwendung von Stichprobendaten betrifft Abfragen mit Spaltenfunktionen wie AVG, SUM und COUNT, bei denen vernünftig genaue Ergebniswerte aus einer Stichprobe der Daten gewonnen werden können. Die Sichprobenerstellung kann auch dazu dienen, eine zufällige Teilmenge der tatsächlichen Zeilen einer Tabelle zu Prüfungszwecken abzurufen oder Data Mining- und Analyseoperationen zu beschleunigen.

DB2 stellt zwei Methoden zur Stichprobenerstellung bereit: Stichproben auf Zeilenebene und Stichproben auf Blockebene.

Stichprobenentnahme auf Zeilenebene nach Bernoulli:

Die Bernoulli-Stichprobe auf Zeilenebene ruft eine Stichprobe von P Prozent der Tabellenzeilen mit Hilfe eines als Suchargument verwendbaren Vergleichselements ab, das jede Zeile mit einer Wahrscheinlichkeit von P/100 einschließt und sie mit einer Wahrscheinlichkeit von 1–P/100 ausschließt.

Die Bernoulli-Stichprobe auf Zeilenebene generiert unabhängig von den Datenwerthäufungen (Clusterbildung) immer eine gültige, zufällige Stichprobe. Allerdings ist die Leistung dieser Stichprobenmethode sehr gering, wenn kein Index verfügbar ist, da jede Zeile abgerufen und das Vergleichselement der Stichprobe auf sie angewandt werden muss. Wenn kein Index vorhanden ist, ergeben sich keine E/A-Einsparungen gegenüber der Ausführung einer Abfrage ohne Stichprobenerstellung. Wenn ein Index vorhanden ist, wird eine bessere Leistung mit dieser Art der Stichprobe erzielt, weil das Vergleichselement der Stichprobe auf die Satz-IDs (Zeilen-ID, RIDs) innerhalb der Indexblattseiten angewandt wird. Im Normalfall erfordert dies nur eine E/A-Operation pro ausgewählter RID und eine E/A-Operation pro Indexblattseite.

Stichprobenentnahme auf Systemseitenebene:

Stichprobe auf Systemseitenebene ist der Stichprobe auf Zeilenebene ähnlich, jedoch werden bei ihr Stichproben der Seiten und nicht der Zeilen erstellt. Eine Seite wird mit einer Wahrscheinlichkeit von P/100 in die Stichprobe eingeschlossen. Das Einschließen einer Seite bedeutet, dass alle Zeilen dieser Seite mit eingeschlossen werden.

Die Leistung der Stichprobenentnahme auf Systemseitenebene ist hervorragend, da nur eine E/A-Operation für jede Seite erforderlich ist, die in die Stichprobe eingeschlossen wird. Im Vergleich zu keiner Stichprobenentnahme verbessert die Stichprobe auf Seitenebene die Leistung um Größenordnungen. Allerdings ist die Genauigkeit kumulierter Schätzung bei Seitenstichproben tendenziell geringer als bei Zeilenstichproben. Diese Diskrepanz in der Genauigkeit wird am deutlichsten, wenn es viele Zeilen pro Block gibt oder die in der Abfrage angegebenen Zeilen einen hohen Grad der Clusterbildung innerhalb der Seiten aufweisen.

Die beste Stichprobenmethode für einen bestimmten Zweck wird durch die Zeitbegrenzung eines Benutzers sowie den gewünschten Grad an Genauigkeit festgelegt.

Angeben der Stichprobenmethode:

Zur Ausführung einer Abfrage an einer zufälligen Stichprobe von Daten aus einer Tabelle können Sie die Klausel TABLESAMPLE der Tabellenverweisklausel in einer SQL-Anweisung verwenden. Zur Angabe der Stichprobenmethode können Sie die Schlüsselwörter BERNOULLI oder SYSTEM verwenden.

Das Schlüsselwort BERNOULLI gibt an, dass eine Bernoulli-Stichprobe auf Zeilenebene auszuführen ist.

Das Schlüsselwort SYSTEM gibt an, dass eine Stichprobe auf Systemseitenebene auszuführen ist, sofern nicht das Optimierungsprogramm feststellt, dass es effizienter ist, stattdessen eine Bernoulli-Stichprobe auf Zeilenebene auszuführen.

Zugehörige Referenzen:

- „Subselect" in *SQL Reference, Volume 1*

Effiziente SELECT-Anweisungen

Da SQL eine flexible Abfragesprache höherer Ebene ist, haben Sie die Möglichkeit, mehrere verschiedene SELECT-Anweisungen zum Abrufen der gleichen Daten zu schreiben. Die Leistung kann jedoch für die verschiedenen Formen der Anweisung sowie für die verschiedenen Optimierungsklassen unterschiedlich sein.

Berücksichtigen die folgenden Richtlinien für SELECT-Anweisungen:

- Geben Sie nur Spalten an, die Sie benötigen. Obwohl es einfacher ist, alle Spalten mit einem Stern (*) anzugeben, führt dies zu überflüssigem Verarbeitungsaufwand und zur Rückgabe unnötiger Spalten.
- Verwenden Sie Vergleichselemente, die die Antwortmenge auf nur die Zeilen einschränken, die Sie benötigen.
- Wenn die Anzahl der Zeilen, die Sie benötigen, bedeutend kleiner ist als die Gesamtanzahl der Zeilen, die abgerufen werden könnten, geben Sie die Klausel

OPTIMIZE FOR an. Diese Klausel wirkt sich sowohl auf die Auswahl der Zugriffspläne als auch auf die Anzahl der im Kommunikationspuffer geblockten Zeilen aus.

- Wenn die Anzahl der abzurufenden Zeilen klein ist, geben Sie nur die Klausel OPTIMIZE FOR *k* ROWS an. Die Klausel FETCH FIRST n ROWS ONLY ist nicht erforderlich. Wenn *n* jedoch groß ist und Sie die ersten *k* Zeilen schnell abrufen wollen und eine mögliche Verzögerung für die nachfolgenden *k* Zeilen in Kauf nehmen, geben Sie beide Klauseln an. Die Größe der Kommunikationspuffer wird in Abhängigkeit vom kleineren der Werte für *n* und *k* festgelegt. Das folgende Beispiel zeigt beide Klauseln:

  ```
  SELECT EMPNAME, SALARY FROM EMPLOYEE
     ORDER BY SALARY DESC
     FETCH FIRST 100 ROWS ONLY
     OPTIMIZE FOR 20 ROWS
  ```

- Zur Verwendung der Zeilenblockung geben Sie die Klausel FOR READ ONLY oder FOR FETCH ONLY an, um die Leistung zu verbessern. Dadurch wird außerdem der gleichzeitige Zugriff verbessert, weil für die abgerufenen Spalten zu keiner Zeit exklusive Sperren aktiviert sind. Zusätzliches Umschreiben der Abfrage kann ebenfalls stattfinden. Durch die Angabe der Klausel FOR READ ONLY oder FOR FETCH ONLY sowie durch die BIND-Option BLOCKING ALL lässt sich die Leistung von Abfragen auf Kurznamen in einem System zusammengeschlossener Datenbanken in ähnlicher Weise verbessern.
- Geben Sie für Cursor, die durch positionierte Aktualisierungen aktualisiert werden, die Klausel FOR UPDATE OF an, um dem Datenbankmanager zu ermöglichen, gleich zu Anfang angemessenere Sperrstufen zu wählen und potenzielle gegenseitige Sperren zu vermeiden. Beachten Sie, dass FOR UPDATE-Cursor die Zeilenblockung nicht nutzen können.
- Für Cursor, die durch Aktualisierungen mit Suche aktualisiert werden, können Sie gegenseitige Sperren vermeiden und trotzdem die Zeilenblockung ermöglichen, indem Sie Sperren des Modus U durch die Klauseln FOR READ ONLY und USE AND KEEP UPDATE LOCKS für die betroffenen Zeilen erzwingen.
- Vermeiden Sie nach Möglichkeit Umsetzungen numerischer Datentypen. Beim Vergleichen von Werten ist es in der Regel effizienter, zwei Werte mit demselben Datentyp miteinander zu vergleichen. Wenn Umwandlungen erforderlich sind, können Ungenauigkeiten aufgrund begrenzter Präzision und Leistungseinbußen aufgrund von Umwandlungen, die zur Laufzeit ausgeführt werden müssen, die Folge sein.

  Verwenden Sie, falls möglich, die folgenden Datentypen:
  - Zeichen (CHAR) anstatt Zeichen variierender Länge (VARCHAR) für kurze Spalten
  - Ganze Zahlen (INTEGER) anstatt Gleitkommazahlen (FLOAT) oder Dezimalzahlen (DECIMAL)
  - Datum/Uhrzeit (Datetime) anstatt Zeichen
  - Numerische Typen anstatt Zeichen
- Lassen Sie zur Verringerung der Wahrscheinlichkeit von Sortieroperationen Klauseln oder Operationen wie DISTINCT oder ORDER BY weg, wenn solche Operationen nicht erforderlich sind.
- Wählen Sie eine einzige Zeile aus, wenn Sie das Vorhandensein von Zeilen prüfen wollen. Öffnen Sie dazu einen Cursor und rufen Sie eine Zeile ab (FETCH) oder führen Sie eine Auswahl für eine Zeile (SELECT INTO) durch. Vergessen Sie nicht, auf den SQLCODE-Wert -811 zu prüfen, wenn mehr als eine Zeile gefunden wird.

  Sofern Sie nicht wissen, dass die Tabelle sehr klein ist, vermeiden Sie die folgende Anweisung zur Prüfung auf einen Nichtnullwert:

```
SELECT COUNT(*) FROM  TABLENAME
```

Das Zählen aller Zeilen wirkt sich bei großen Tabellen negativ auf die Leistung aus.

- Wenn das Aktualisierungsaufkommen gering ist und die Tabellen umfangreich sind, sollten Sie Indizes für Spalten definieren, die häufig in Vergleichselementen verwendet werden.
- Ziehen Sie die Verwendung einer IN-Liste in Betracht, wenn dieselbe Spalte in mehreren PREDICATE-Klauseln erscheint. Bei großen IN-Listen, die mit Hostvariablen verwendet werden, können Verarbeitungsschleifen für Untergruppen der Hostvariablen die Leistung verbessern.

Die folgenden Vorschläge gelten insbesondere für SELECT-Anweisungen, die auf mehrere Tabellen zugreifen.

- Verwenden Sie Verknüpfungsvergleichselemente zur Verknüpfung von Tabellen. Ein Verknüpfungsvergleichselement ist ein Vergleich zwischen zwei Spalten verschiedener Tabellen in einer Verknüpfung.
- Definieren Sie Indizes für die Spalten in dem Verknüpfungsvergleichselement, um eine effizientere Verarbeitung der Verknüpfung zu ermöglichen. Indizes sind außerdem für UPDATE- und DELETE-Anweisungen förderlich, die SELECT-Anweisungen enthalten, die auf mehrere Tabellen zugreifen.
- Vermeiden Sie nach Möglichkeit Ausdrücke oder Klauseln OR mit Verknüpfungsvergleichselementen, weil der Datenbankmanager einige Verknüpfungstechniken nicht nutzen kann. Dies kann dazu führen, dass nicht die effizienteste Verknüpfungsmethode gewählt wird.
- In einer Umgebung mit partitionierten Datenbanken stellen Sie nach Möglichkeit sicher, dass beide Tabellen, die verknüpft werden, über die Verknüpfungsspalte partitioniert sind.

Zugehörige Konzepte:

- „Richtlinien zur Begrenzung von SELECT-Anweisungen" auf Seite 90
- „Richtlinien zur Abfrageoptimierung" auf Seite 95

Richtlinien für Compound-SQL-Anweisungen

Zur Verringerung des Systemaufwands für den Datenbankmanager können Sie mehrere SQL-Anweisungen zu einem einzigen ausführbaren Anweisungsblock zusammenfassen. Da die SQL-Anweisungen in diesem Block Unteranweisungen sind, die einzeln ausgeführt werden könnten, wird diese Art von Code als *Compound-SQL* (Verbund-SQL) bezeichnet. Neben der Verringerung des Datenbankmanageraufwands reduzieren Compound-SQL-Anweisungen für ferne Clients zudem die Anzahl der Anforderungen, die über das Netz übertragen werden müssen.

Es gibt zwei Arten von Compound-SQL-Anweisungen:

- **Ganzheitliche Compound-SQL-Anweisungen**

  Die Anwendung empfängt eine Antwort vom Datenbankmanager, wenn alle Unteranweisungen erfolgreich ausgeführt wurden oder wenn eine Unteranweisung mit einem Fehler beendet wurde. Wenn eine Unteranweisung mit einem Fehler beendet wurde, wird der gesamte Block als fehlerhaft angesehen. Alle Änderungen, die innerhalb des Blocks an der Datenbank vorgenommen wurden, werden rückgängig gemacht.

Ganzheitliche Compound-SQL-Anweisungen (Atomic Compound SQL) werden von DB2 Connect nicht unterstützt.

- **Nicht ganzheitliche Compound-SQL-Anweisungen**

  Die Anwendung empfängt eine Antwort vom Datenbankmanager, wenn alle Unteranweisungen ausgeführt wurden. Es werden alle Unteranweisungen innerhalb eines Blocks ausgeführt ohne Rücksicht darauf, ob die zuvor ausgeführte Unteranweisung erfolgreich beendet wurde oder nicht. Der Block von Anweisungen kann nur rückgängig gemacht werden, wenn die Arbeitseinheit, die diese nicht ganzheitliche Compound-SQL-Anweisung enthält, rückgängig gemacht wird.

Compound-SQL-Anweisungen werden in gespeicherten Prozeduren (auch als DARI-Routinen bezeichnet) sowie in den folgenden Prozessen zur Anwendungsentwicklung unterstützt:

- Eingebettetes statisches SQL
- DB2 Call Level Interface
- JDBC

Dynamische Compound-SQL-Anweisungen

Dynamische Compound-Anweisungen werden von DB2® als einzelne Anweisung kompiliert. Eine solche Anweisung lässt sich effektiv für kurze Prozeduren einsetzen, die nur wenig Steuerungsflusslogik, jedoch einen beträchtlichen Datenfluss erfordern. Ziehen Sie für größere Konstrukte mit komplexem verschachteltem Steuerungsfluss die Verwendung von SQL-Prozeduren in Betracht.

In einer dynamischen Compound-Anweisung können Sie die folgenden Elemente in Deklarationen verwenden:

- SQL-Variablen in Variablendeklarationen von Unterweisungen
- Bedingungen in den Unteranweisungen auf der Basis der SQLSTATE-Werte der Bedingungsdeklaration
- Eine oder mehrere SQL-Prozeduranweisungen

Dynamische Compound-Anweisungen können außerdem verschiedene Anweisungen zur Steuerung des logischen Ablaufs wie FOR, IF, ITERATE und WHILE enthalten.

Wenn in einer dynamischen Compound-Anweisung ein Fehler auftritt, werden alle vorhergehenden SQL-Anweisungen rückgängig gemacht und die verbleibenden SQL-Anweisungen in der dynamischen Compound-Anweisung werden nicht verarbeitet.

Eine dynamische Compound-Anweisung kann in einen Auslöser, eine SQL-Funktion bzw. eine SQL-Methode eingebettet oder durch dynamische SQL-Anweisungen abgesetzt werden. Eine solche ausführbare Anweisung kann dynamisch vorbereitet werden. Der Aufruf der Anweisung erfordert keine Zugriffsrechte, allerdings muss die der Anweisung zugeordnete Berechtigungs-ID über die erforderlichen Zugriffsrechte für den Aufruf der SQL-Anweisungen in der Compound-Anweisung verfügen.

Zugehörige Konzepte:

- „Richtlinien zur Abfrageoptimierung” auf Seite 95

Richtlinien zur Zeichenkonvertierung

Eine Datenkonvertierung kann erforderlich sein, um Daten zwischen den Codepages einer Anwendung und der Datenbank zuzuordnen, wenn die Anwendung und die Datenbank nicht mit der gleichen Codepage arbeiten. Da die Zuordnung und die Datenkonvertierung zusätzlichen Systemaufwand verursachen, verbessert sich die Anwendungsleistung, wenn die Anwendung und die Datenbank die gleiche Codepage oder die gleiche Identitätssortierfolge verwenden.

Eine Zeichenkonvertierung kann in folgenden Fällen stattfinden:

- Wenn ein Client oder eine Anwendung mit einer anderen Codepage ausgeführt wird als die Datenbank, auf die zugegriffen wird.

  Die Konvertierung findet auf der Datenbankservermaschine statt, die die Daten empfängt. Wenn der Datenbankserver die Daten empfängt, erfolgt die Zeichenkonvertierung aus der Codepage der Anwendung in die Codepage der Datenbank. Wenn die Anwendungsmaschine die Daten empfängt, erfolgt die Konvertierung aus der Codepage der Datenbank in die Codepage der Anwendung.
- Wenn ein Client oder eine Anwendung, die eine Datei importiert oder lädt, mit einer anderen Codepage arbeitet als die Datei, die importiert oder geladen wird.

Für die folgenden Objekten findet keine Zeichenkonvertierung statt:

- Dateinamen.
- Daten, die aus einer Spalte mit dem Attribut FOR BIT DATA stammen oder dafür vorgesehen sind, oder Daten, die in einer SQL-Operation verwendet werden, deren Ergebnis Daten mit dem Attribut FOR BIT- bzw. BLOB-Daten sind.
- Ein DB2®-Produkt oder eine Plattform, für die keine unterstützte Konvertierungsfunktion für EUC oder UCS-2 installiert ist. Ihre Anwendung empfängt in diesem Fall einen Fehler mit dem SQLCODE-Wert -332 (SQLSTATE 57017).

Die Konvertierungsfunktion und die Konvertierungstabellen bzw. die DBCS-Konvertierungs-APIs, die der Datenbankmanager zur Konvertierung von Mehrbytecodepages verwendet, hängen von der Betriebssystemumgebung ab.

Anmerkung: Zeichenfolgenkonvertierungen zwischen Mehrbytecodepages, zum Beispiel zwischen DBCS und EUC, können zu einer Verkürzung bzw. zu einer Verlängerung von Zeichenfolgen führen. Außerdem können Codepunkte, die verschiedenen Zeichen in den codierten PC-Zeichensätzen für DBCS, EUC und UCS-2 zugeordnet sind, zu unterschiedlichen Ergebnissen bei der Sortierung derselben Zeichen führen.

Codepageunterstützung für Erweiterten UNIX®-Code (EUC)

Hostvariablen, die Grafikdaten in C- oder C++-Anwendungen verwenden, bedürfen einiger besonderer Überlegungen, zu denen spezielle Aspekte des Precompilers, der Leistung von Anwendungen und des Anwendungsentwurfs gehören.

Viele Zeichen in den EUC-Codepages für Japanisch und traditionelles Chinesisch erfordern besondere Methoden zur Verwaltung der Unterstützung für Datenbanken und Clientanwendungen für Grafikdaten, die Doppelbytezeichen erfordern. Grafikdaten aus diesen EUC-Codepages werden mit Hilfe des codierten UCS-2-Zeichensatzes gespeichert und bearbeitet.

Zugehörige Konzepte:

- „Richtlinien zur Analyse, wo eine Abfrage für zusammengeschlossene Datenbanken ausgewertet wird" auf Seite 215

Zugehörige Referenzen:

- „Konvertierungstabellen für die Codepages 923 und 924" in *Systemverwaltung: Konzept*
- „Konvertierungstabellendateien für Euro-fähige Codepages" in *Systemverwaltung: Konzept*

Richtlinien für gespeicherte Prozeduren

Gespeicherte Prozeduren ermöglichen es, durch einen einmaligen Aufruf an eine ferne Datenbank eine vorprogrammierte Prozedur in einer Datenbankanwendungsumgebung auszuführen, in der viele Situationen wiederholt auftreten. Zum Beispiel könnten der Empfang einer festen Menge von Daten, Durchführung derselben Folge von Anforderungen für eine Datenbank oder das Zurückgeben einer festen Menge von Daten mehrere Zugriffe auf die Datenbank darstellen.

Für die Verarbeitung einer einzigen SQL-Anweisung für eine ferne Datenbank sind zwei Übertragungen erforderlich: eine Anforderungsübertragung und eine Empfangsübertragung. Da eine Anwendung viele SQL-Anweisungen enthält, macht sie viele Übertragungen zur Ausführung ihrer Arbeit erforderlich.

Wenn ein Datenbankclient jedoch eine gespeicherte Prozedur verwendet, in die viele SQL-Anweisungen eingebunden sind, sind nur zwei Übertragungen für den gesamten Prozess erforderlich.

Gespeicherte Prozeduren werden in der Regel in Prozessen ausgeführt, die von den Datenbankagenten getrennt sind. Diese Trennung macht es erforderlich, dass die Prozesse der gespeicherten Prozeduren und die Agentenprozesse über einen Router miteinander kommunizieren. Allerdings kann eine spezielle Art von gespeicherter Prozedur, die im Agentenprozess ausgeführt wird, die Leistung verbessern, wenngleich dies ein erhebliches Risiko mit sich bringt, dass Daten und Datenbanken beschädigt werden könnten.

Diese gefährlichen gespeicherten Prozeduren sind diejenigen, die als *nicht abgeschirmt* (not fenced) erstellt werden. Bei einer nicht abgeschirmten Prozedur gibt es zwischen der gespeicherten Prozedur und den von dem Datenbankagenten genutzten Datenbanksteuerstrukturen keine Trennung. Wenn ein Datenbankadministrator (DBA) sicherstellen will, dass die Operationen einer gespeicherten Prozedur nicht versehentlich oder böswillig Datenbanksteuerstrukturen beschädigen, kommt die Option *nicht abgeschirmter* Prozeduren nicht in Frage.

Verwenden Sie angesichts des vorhandenen Risikos einer Datenbankbeschädigung *nicht abgeschirmte* gespeicherte Prozeduren **nur**, wenn Sie die maximalen Leistungsvorteile erzielen müssen. Stellen Sie darüber hinaus absolut sicher, dass die Prozedur einwandfrei codiert ist und gründlich getestet wurde, bevor Sie sie für die Ausführung als nicht abgeschirmte gespeicherte Prozedur zulassen. Wenn es zu einem schwerwiegenden Fehler während der Ausführung einer nicht abgeschirmten Prozedur kommt, stellt der Datenbankmanager fest, ob der Fehler in der Anwendung oder im Code des Datenbankmanagers auftrat, und führt die entsprechenden Wiederherstellungsmaßnahmen aus.

Eine nicht abgeschirmte gespeicherte Prozedur kann den Datenbankmanager so beschädigen, dass dieser nicht wiederhergestellt werden kann. Dies kann zu Datenverlust und zu einer beschädigten Datenbank führen. Gehen Sie äußerst vorsichtig vor, wenn Sie nicht abgeschirmte vertrauenswürdige gespeicherte Prozeduren ausführen. In nahezu allen Fällen führt die richtige Leistungsanalyse einer Anwendung zu einer guten Leistung, ohne nicht abgeschirmte gespeicherte Prozeduren verwenden zu müssen. Zum Beispiel können Auslöser die Leistung verbessern.

Zugehörige Konzepte:

• „Richtlinien zur Abfrageoptimierung" auf Seite 95

Parallelverarbeitung für Anwendungen

DB2® unterstützt parallele Umgebungen in erster Linie auf symmetrischen Multiprozessormaschinen (SMP-Maschinen), jedoch in begrenztem Maß auch auf Einzelprozessormaschinen. In SMP-Maschinen kann mehr als ein Prozessor auf die Datenbank zugreifen, so dass komplexe SQL-Anforderungen zur parallelen Ausführung unter den Prozessoren aufgeteilt werden können.

Verwenden Sie zur Angabe des Grades der Parallelität, die bei der Kompilierung einer Anwendung implementiert werden soll, das Sonderregister CURRENT DEGREE oder die Bindeoption DEGREE. *Grad* bezieht sich in diesem Zusammenhang auf die Anzahl der Teile einer Abfrage, die gleichzeitig ausgeführt werden. Es gibt keine strenge Beziehung zwischen der Anzahl der Prozessoren und dem Wert, den Sie für den Grad der Parallelität auswählen. Sie können einen Wert angeben, der größer oder kleiner ist als die Anzahl von Prozessoren auf der Maschine. Selbst für Einzelprozessormaschinen können Sie einen höheren Grad als 1 definieren, um die Leistung auf die eine oder andere Art zu verbessern. Beachten Sie jedoch, das jeder weitere Grad an Parallelität den Hauptspeicherbedarf und die CPU-Belastung im System erhöht.

Einige Konfigurationsparameter müssen modifiziert werden, um die Leistung zu optimieren, wenn Abfragen parallel ausgeführt werden. Insbesondere für eine Umgebung mit einem hohen Grad an Parallelität sollten Sie die Konfigurationsparameter prüfen und modifizieren, über die die Größen des gemeinsamen Speichers und des Vorablesezugriffs gesteuert werden.

Die folgenden drei Konfigurationsparameter steuern und verwalten die partitionsinterne Parallelität:

- Der Konfigurationsparameter *intra_parallel* des Datenbankmanagers aktiviert oder inaktiviert die Unterstützung für Parallelverarbeitung.
- Der Datenbankkonfigurationsparameter *max_querydegree* definiert eine obere Grenze für den Grad der Parallelität für alle Abfragen in der Datenbank. Dieser Wert hat Vorrang vor dem Wert des Sonderregisters CURRENT DEGREE und der Bindeoption DEGREE.
- Der Datenbankkonfigurationsparameter *dft_degree* definiert den Standardwert für das Sonderregister CURRENT DEGREE und die Bindeoption DEGREE.

Wenn eine Abfrage mit der Definition DEGREE = ANY kompiliert wird, wählt der Datenbankmanager den Grad der partitionsinternen Parallelität anhand einer Reihe von Faktoren aus, zu denen die Anzahl der Prozessoren und die Merkmale der Abfrage gehören. Der tatsächlich bei der Ausführung verwendete Grad kann aufgrund dieser Faktoren und des Aktivitätenaufkommens im System niedriger als die Anzahl der Prozessoren sein. Der Grad der Parallelität kann vor der Abfrageausführung gesenkt werden, wenn das System sehr ausgelastet ist. Der Grund hierfür

liegt in der intensiven Nutzung der Systemressourcen durch die partitionsinterne Parallelität, die einerseits die abgelaufene Zeit für eine Abfrage verringert, sich andererseits jedoch auch negativ auf die Leistung für andere Datenbankbenutzer auswirken kann.

Verwenden Sie die SQL-EXPLAIN-Einrichtung zum Anzeigen des Zugriffsplans, wenn Sie Informationen über den vom Optimierungsprogramm ausgewählten Grad der Parallelität erhalten wollen. Zum Anzeigen von Informationen über den tatsächlich bei der Ausführung genutzten Grad der Parallelität verwenden Sie den Datenbanksystemmonitor.

Parallelität in Umgebungen ohne SMP-Maschinen

Sie können einen Grad an Parallelität angeben, ohne eine SMP-Maschine zu haben. Zum Beispiel können ein-/ausgabegebundene Abfragen auf einer Einzelprozessormaschine von der Deklaration eines Grades 2 oder höher profitieren. In diesem Fall braucht der Prozessor vielleicht nicht auf die Beendigung von Eingabe- bzw. Ausgabefunktionen zu warten, bevor er mit der Verarbeitung einer neuen Abfrage beginnt. Die Deklaration eines Grades 2 oder höher steuert die E/A-Parallelität auf einer Einzelprozessormaschine jedoch nicht direkt. Dienstprogramme wie Load können die E/A-Parallelität unabhängig von einer solchen Deklaration steuern. Das Schlüsselwort `ANY` kann ebenfalls dazu verwendet werden, den Konfigurationsparameter *dft_degree* des Datenbankmanagers zu definieren. Das Schlüsselwort `ANY` gibt dem Optimierungsprogramm die Möglichkeit, den Grad der partitionsinternen Parallelität zu bestimmen.

Zugehörige Konzepte:

- „EXPLAIN-Tools" auf Seite 224
- „Optimierungsstrategien für partitionsinterne Parallelität" auf Seite 203

Zugehörige Referenzen:

- „max_querydegree - Maximaler Grad der Parallelität bei Abfragen" auf Seite 522
- „intra_parallel - Partitionsinterne Parallelität aktivieren" auf Seite 522
- „dft_degree - Standardgrad der Parallelität" auf Seite 502

Verbessern der Leistung durch Binden mit REOPT

SQL-Abfragen können bei der Ausführung eine schlechte Leistung zeigen, wenn die Werte, die für die Eingabevariablen, wie zum Beispiel Parametermarken, Hostvariablen und Sonderregister, verwendet werden, außerhalb des vorhersagbaren Bereichs der Schätzwerte für Standardfilterfaktoren liegen. Standardfilterfaktoren, die in Szenarios verwendet werden, in denen der tatsächliche Datenwert unbekannt ist, sind Schätzwerte für die Anzahl von Zeilen, die tatsächlich zur Laufzeit zurückgegeben werden, wenn der tatsächliche Datenwert verwendet wird.

Die Bindeoption REOPT gibt an, ob DB2® einen Zugriffspfad unter Verwendung von Werten für Hostvariablen, Parametermarken und Sonderregistern optimieren soll. REOPT-Werte werden durch die folgenden Argumente in den Befehlen BIND, PREP und REBIND angegeben:

REOPT NONE
Der Zugriffspfad für eine gegebene SQL-Anweisung, in der Hostvariablen, Parametermarken oder Sonderregister enthalten sind, wird nicht mit realen Werten für diese Variablen optimiert. Stattdessen werden die Standardschätzwerte für diese Variablen verwendet. Ein solcher Plan wird in den Cache gestellt und nachfolgend verwendet. Dies ist das Standardverhalten.

REOPT ONCE
Der Zugriffspfad für eine gegebene SQL-Anweisung wird mit den realen Werten der Hostvariablen, Parametermarken oder Sonderregister optimiert, wenn die Abfrage zum ersten Mal ausgeführt wird. Ein solcher Plan wird in den Cache gestellt und nachfolgend verwendet.

REOPT ALWAYS
Der Zugriffspfad für eine gegebene SQL-Anweisung wird immer kompiliert und mit den Werten der Hostvariablen, Parametermarken oder Sonderregister reoptimiert, die bei der jeweiligen Ausführung bekannt sind.

Zugehörige Konzepte:

- „Effects of REOPT on static SQL" in *Application Development Guide: Programming Client Applications*
- „Effects of REOPT on dynamic SQL" in *Application Development Guide: Programming Client Applications*

Kapitel 4. Überlegungen zur Umgebung

Neben den Faktoren, die Sie beim Entwurf und bei der Codierung Ihrer Anwendung berücksichtigen (siehe Kapitel 3, „Überlegungen zu Anwendungen", auf Seite 45), können auch bestimmte Faktoren der Umgebung die Auswahl des Zugriffsplans beeinflussen.

Weitere Informationen zu Faktoren, die sich direkt auf das SQL-Optimierungsprogramm auswirken, finden Sie in Kapitel 5, „Systemkatalogstatistiken", auf Seite 113.

Wenn Sie Anwendungen optimieren und Änderungen an der Umgebung vornehmen, müssen Sie Ihre Anwendungen erneut binden, um sicherzustellen, dass der beste Zugriffsplan verwendet wird.

Auswirkung von Datenbankpartitionsgruppen auf die Abfrageoptimierung

In partitionierten Datenbanken erkennt das Optimierungsprogramm die Kollokation von Tabellen und nutzt sie bei der Bestimmung des besten Zugriffsplans für eine Abfrage. Wenn Tabellen häufig in Verknüpfungsabfragen einbezogen werden, sollten sie auf Partitionen in einer partitionierten Datenbank aufgeteilt werden, so dass sich die Zeilen aus jeder Tabelle, die verknüpft wird, in der gleichen Datenbankpartition befinden. Während der Verknüpfungsoperation wird durch die Kollokation der Daten aus beiden verknüpften Tabellen eine Verschiebung der Daten von einer Partition in die andere vermieden. Speichern Sie beide Tabellen in derselben Datenbankpartitionsgruppe, um sicherzustellen, dass die Daten aus den Tabellen durch Kollokation zusammengefasst werden.

In einer partitionierten Datenbank wird durch das Verteilen von Daten auf mehrere Partitionen abhängig von der Größe der Tabelle die geschätzte Dauer (bzw. der Aufwand) zur Ausführung einer Abfrage verringert. Die Anzahl der Tabellen, die Größe der Tabellen, die Speicherposition der Daten in diesen Tabellen und die Art der Abfrage (d. h., ob eine Verknüpfung erforderlich ist) wirken sich alle auf den Aufwand für die Abfrage aus.

Zugehörige Konzepte:

- „Verknüpfungsstrategien in partitionierten Datenbanken" auf Seite 193
- „Verknüpfungsmethoden in partitionierten Datenbanken" auf Seite 195
- „Partitionen in einer partitionierten Datenbank" auf Seite 332

Auswirkung von Tabellenbereichen auf die Abfrageoptimierung

Bestimmte Merkmale der verwendeten Tabellenbereiche können die Auswahl des Zugriffsplans durch den SQL-Compiler beeinflussen:

- Kenndaten der Behälter

  Behälterkenndaten können sich wesentlich auf den Ein-/Ausgabeaufwand auswirken, der mit der Abfrageausführung verbunden ist. Bei der Auswahl eines Zugriffsplans berücksichtigt das SQL-Optimierungsprogramm diesen Ein-/Ausgabeaufwand, einschließlich aller Unterschiede in den Aufwänden für die

Zugriffe auf Daten verschiedener Tabellenbereiche. Zwei Spalten im Systemkatalog SYSCAT.TABLESPACES werden vom Optimierungsprogramm zur Abschätzung der E/A-Aufwände für den Zugriff auf Daten in einem Tabellenbereich herangezogen:

– Die Spalte OVERHEAD, die einen Schätzwert in Millisekunden für die Zeit enthält, die der Behälter benötigt, bevor irgendwelche Daten in den Speicher gelesen werden. In diesen Wert fließen der Aufwand für den E/A-Controller des Behälters und die Latenzzeit der Platte, zur der auch die Suchzeit der Platte gehört, mit ein.

  Mit Hilfe der folgenden Formel lässt sich der Systemaufwand (OVERHEAD) abschätzen:

  ```
  OVERHEAD = durchschnittliche Suchzeit in Millisekunden
             + (0,5 * rotationsbedingte Latenzzeit)
  ```

  Dabei gilt Folgendes:
  - 0,5 stellt den durchschnittlichen Aufwand für eine halbe Umdrehung (Rotation) dar.
  - Die rotationsbedingte Latenzzeit für jede vollständige Umdrehung wird wie folgt in Millisekunden berechnet:

    ```
    (1 / Umdrehung pro Minute) * 60 * 1000
    ```

    Dabei sind die Berechnungsschritte wie folgt:
    • Dividieren durch die Umdrehungen pro Minute, um die Minuten pro Umdrehung zu erhalten
    • Multiplizieren mit 60, um die Sekunden pro Umdrehung zu erhalten
    • Multiplizieren mit 1000, um die Millisekunden pro Umdrehung zu erhalten

  Als Beispiel sei eine Plattengeschwindigkeit von 7 200 Umdrehungen pro Minute angenommen. Mit Hilfe der Umdrehungs-Latenz-Formel ergäbe sich folgender Wert:

  ```
  (1 / 7200) * 60 * 1000 = 8,328 Millisekunden
  ```

  Das Ergebnis kann anschließend in der Berechnung des Schätzwerts für OVERHEAD bei einer angenommenen durchschnittlichen Suchzeit von 11 Millisekunden verwendet werden:

  ```
  OVERHEAD = 11 + (0,5 * 8,328)
           = 15,164
  ```

  Der geschätzte OVERHEAD-Wert läge in diesem Fall bei ca. 15 Millisekunden.

– Die Spalte TRANSFERRATE, die einen Schätzwert in Millisekunden für die Zeit enthält, die zum Einlesen einer Datenseite in den Hauptspeicher benötigt wird.

  Wenn jeder Tabellenbereichsbehälter eine einzelne physische Platte ist, können Sie mit Hilfe der folgenden Formel den Übertragungsaufwand pro Seite in Millisekunden abschätzen:

  ```
  TRANSFERRATE = (1 / spec_rate) * 1000 / 1 024 000 * seitengröße
  ```

  Dabei gilt folgendes:
  - `spec_rate` ist die Übertragungsgeschwindigkeit in MB pro Sekunde, die für die Platte angegeben wird.
  - Dividieren durch `spec_rate`, um die Sekunden pro MB zu erhalten
  - Multiplizieren mit 1000, um die Millisekunden pro MB zu erhalten
  - Dividieren durch 1 024 000, um die Millisekunden pro Byte zu erhalten

- Multiplizieren mit der Seitengröße in Byte (z. B. 4 096 Byte für eine 4-KB-Seite), um die Millisekunden pro Seite zu erhalten

Als Beispiel sei eine angegebene Übertragungsgeschwindigkeit (spec_rate) von 3 MB pro Sekunde angenommen. Daraus ergäbe sich folgende Berechnung:

```
TRANSFERRATE = (1 / 3) * 1000 / 1024000 * 4096
             = 1,333248
```

Der geschätzte TRANSFERRATE-Wert läge hier bei ca. 1,3 Millisekunden pro Seite.

Wenn es sich bei den Tabellenbereichsbehältern nicht um einzelne physische Platten, sondern um Platteneinheiten (Disk-Arrays, z. B. RAID) handelt, sind zusätzliche Punkte bei der Abschätzung des zu verwendenden Werts für TRANSFERRATE beachten. Wenn die Platteneinheit relativ klein ist, können Sie den Wert für `spec_rate` mit der Anzahl Platten multiplizieren, da anzunehmen ist, dass der Engpass auf Plattenebene liegt.

Ist die Anzahl der Platten in der Einheit, die den Behälter bildet, jedoch groß, liegt der Engpass vielleicht nicht auf Plattenebene, sondern bei einer der E/A-Subsystemkomponenten wie Einheitencontroller, E/A-Busse oder dem Systembus. In diesem Fall kann nicht angenommen werden, dass der E/A-Durchsatz das Produkt aus `spec_rate` und der Anzahl der Platten ist. Stattdessen muss die tatsächliche E/A-Geschwindigkeit während einer sequenziellen Tabellensuche in MB gemessen werden. Zum Beispiel könnte eine Tabellensuche mit einer Anweisung wie `select count(*) from große_tabelle` durchgeführt werden, die mehrere MB umfasst. Dividieren Sie das Ergebnis durch die Anzahl der Behälter, die den Tabellenbereich bilden, in dem `große_tabelle` gespeichert ist. Setzen Sie das Ergebnis für `spec_rate` in die oben angegebene Formel ein. Zum Beispiel würde eine gemessene sequenzielle E/A-Geschwindigkeit von 100 MB beim Durchsuchen einer Tabelle in einem Tabellenbereich mit vier Behältern einen Wert von 25 MB pro Behälter bzw. einen TRANSFERRATE-Wert von (1/25) * 1000 / 1024000 * 4096 = 0,16 Millisekunden pro Seite bedeuten.

Jeder der einem Tabellenbereich zugeordneten Behälter kann sich auf einer anderen physischen Platte befinden. Um die besten Ergebnisse erzielen zu können, sollten alle physischen Platten, die für einen bestimmten Tabellenbereich verwendet werden, über die gleichen Werte für OVERHEAD und TRANSFERRATE verfügen. Wenn diese Merkmale nicht übereinstimmen, sollten Sie bei der Einstellung der Werte für OVERHEAD und TRANSFERRATE jeweils den Mittelwert verwenden.

Medienspezifische Werte für diese Spalten können Sie den technischen Daten zur Hardware entnehmen oder durch Experimentieren ermitteln. Diese Werte können in den Anweisungen CREATE TABLESPACE und ALTER TABLESPACE angegeben werden.

Experimentieren wird in der oben erwähnten Umgebung, in der eine Platteneinheit als Behälter eingesetzt wird, besonders wichtig. Es empfiehlt sich, eine einfache Abfrage, die Daten versetzt, zu erstellen und sie in Verbindung mit einem plattformspezifischen Messprogramm auszuführen. Anschließend können Sie die Abfrage mit anderen Behälterkonfigurationen innerhalb des Tabellenbereichs wiederholen. Mit Hilfe der Anweisungen CREATE und ALTER TABLESPACE können Sie die Art und Weise ändern, wie Daten in der Umgebung übertragen werden.

Die durch diese beiden Werte bereitgestellten Informationen zum Ein-/Ausgabeaufwand können das Optimierungsprogramm in mehrfacher Weise beeinflussen, zum Beispiel dahin gehend, ob ein Index für den Zugriff auf die Daten zu verwenden ist und welche Tabellen als innere und äußere Tabellen in einer Verknüpfung auszuwählen sind.

- Vorablesezugriff

  Bei der Kalkulation des Ein-/Ausgabeaufwands für den Zugriff auf Daten in einem Tabellenbereich berücksichtigt das Optimierungsprogramm auch die potenziellen Auswirkungen, die das Vorablesen von Daten- und Indexseiten von Platten auf die Leistung der Abfrage haben kann. Der Vorablesezugriff auf Daten- und Indexseiten kann den Aufwand und die Wartezeit verringern, die mit dem Einlesen der Daten in den Pufferpool verbunden sind.

  Das Optimierungsprogramm verwendet die Informationen der Spalten PREFETCHSIZE und EXTENTSIZE im Systemkatalog SYSCAT.TABLESPACES, um die Menge der durch den Vorablesezugriff gelesenen Daten für einen Tabellenbereich abzuschätzen.

  – Der Wert für EXTENTSIZE kann nur bei der Erstellung eines Tabellenbereichs (z. B. mit der Anweisung CREATE TABLESPACE) festgelegt werden. Der Standardwert für EXTENTSIZE beträgt 32 Seiten (von jeweils 4 KB) und ist in der Regel ausreichend.
  – Der Wert für PREFETCHSIZE kann sowohl bei der Erstellung als auch bei der Änderung des Tabellenbereichs mit der Anweisung ALTER TABLESPACE festgelegt werden. Der Standardwert für PREFETCHSIZE wird durch den Wert des Konfigurationsparameters DFT_PREFETCH_SZ der Datenbank festgelegt, der je nach Betriebssystem unterschiedlich ist. Lesen Sie die Empfehlungen zur Einstellung dieses Parameters und nehmen Sie Änderungen nach Bedarf vor, um das Versetzen von Daten zu verbessern.

Das folgende Beispiel zeigt die Syntax zur Änderung der Merkmale des Tabellenbereichs RESOURCE:

```
ALTER TABLESPACE RESOURCE
  PREFETCHSIZE 64
  OVERHEAD     19.3
  TRANSFERRATE 0.9
```

Ziehen Sie nach Durchführung von Änderungen an den Tabellenbereichen in Betracht, die Anwendungen erneut zu binden und das Dienstprogramm RUNSTATS zum Erfassen der aktuellsten Statistikdaten zu Indizes auszuführen, um sicherzustellen, dass die besten Zugriffspläne verwendet werden.

Zugehörige Konzepte:

- „Katalogstatistiktabellen" auf Seite 126
- „Der SQL-Compilerprozess" auf Seite 157
- „Illustration der Adressenzuordnung eines DMS-Tabellenbereichs" auf Seite 19

Serveroptionen mit Einfluss auf zusammengeschlossene Datenbanken

Ein System mit zusammengeschlossenen Datenbanken (Federated System) besteht aus einem DB2®-Datenbankverwaltungssystem (DBMS), d. h. der zusammengeschlossenen Datenbank, und einer oder mehreren Datenquellen. Sie geben die Datenquellen für die zusammengeschlossene Datenbank an, wenn Sie Anweisungen CREATE SERVER absetzen. Beim Absetzen dieser Anweisungen können Sie Serveroptionen angeben, die bestimmte Aspekte des Betriebs des Systems mit der zusammengeschlossenen Datenbank bezüglich DB2 und der angegebenen Datenquelle eingrenzen und steuern. Wenn Sie später Serveroptionen ändern wollen, verwenden Sie Anweisungen ALTER SERVER.

Anmerkung: Sie müssen die Installationsoption für verteilte Verknüpfung (*distributed join*) installieren und den Parameter FEDERATED des Datenbankmanagers auf den Wert YES setzen, bevor Sie Server erstellen und Serveroptionen angeben können.

Die Werte der Serveroptionen, die Sie angeben, haben Einfluss auf die Pushdown-Analyse von Abfragen, die globale Optimierung und andere Aspekte von Operationen mit zusammengeschlossenen Datenbanken. Zum Beispiel können Sie in der Anweisung CREATE SERVER Leistungsstatistikdaten als Serveroptionswerte angeben, wie zum Beispiel die Option *cpu_ratio*, mit der die relativen Geschwindigkeiten der CPUs an der Datenquelle und auf dem Server der zusammengeschlossenen Datenbank angegeben werden. Sie könnten auch die Option *io_ratio* auf einen Wert setzen, der die relativen Übertragungsgeschwindigkeiten der E/A-Einheiten der Quelle und des Servers der zusammengeschlossenen Datenbank angibt. Bei der Ausführung der Anweisung CREATE SERVER werden diese Daten der Katalogsicht SYSCAT.SERVEROPTIONS hinzugefügt, und das Optimierungsprogramm bezieht sie in die Entwicklung eines Zugriffsplans für die Datenquelle mit ein. Falls sich ein Statistikwert ändert (wie es beispielsweise bei einer Aufrüstung der CPU der Datenquelle möglich ist), können Sie die Katalogsicht SYSCAT.SERVEROPTIONS mit Hilfe der Anweisung ALTER SERVER mit dieser Änderung aktualisieren. Das Optimierungsprogramm verwendet dann die neuen Informationen beim nächsten Auswählen eines Zugriffsplans für die Datenquelle.

Zugehörige Referenzen:

- „ALTER SERVER statement" in *SQL Reference, Volume 2*
- „SYSCAT.SERVEROPTIONS catalog view" in *SQL Reference, Volume 1*
- „federated - Unterstützung für Systeme mit zusammengeschlossenen Datenbanken" auf Seite 532
- „CREATE SEQUENCE statement" in *SQL Reference, Volume 2*

Kapitel 5. Systemkatalogstatistiken

In den Systemkatalogen gespeicherte statistische Daten helfen dem Optimierungsprogramm bei der Auswahl des besten Zugriffsplans für Abfragen. Stellen Sie sicher, dass Sie das Dienstprogramm RUNSTATS wie folgt ausführen, um diese statistischen Daten zu aktualisieren:

- In regelmäßigen Intervallen für Tabellen, deren Inhalt sich kontinuierlich ändert.
- Nach jeder Operation, die Daten in einer bedeutenden Anzahl von Tabellenzeilen hinzufügt bzw. ändert. Solche Operationen sind beispielsweise im Stapelbetrieb durchgeführte Aktualisierungen und Datenladeoperationen (LOAD), die Zeilen hinzufügen.

Katalogstatistiken

Wenn der SQL-Compiler SQL-Abfragepläne optimiert, werden die Entscheidungen in hohem Maße von statistischen Informationen über die Größe der Tabellen und Indizes der Datenbank beeinflusst. Das Optimierungsprogramm verwendet außerdem Informationen über die Verteilung von Daten in bestimmten Spalten von Tabellen und Indizes, sofern diese Spalten zur Auswahl von Spalten oder Verknüpfung von Tabellen herangezogen werden. Das Optimierungsprogramm schätzt anhand dieser Informationen den Aufwand für alternative Zugriffspläne für jede Abfrage ab.

Über die Informationen zu Tabellengrößen und Datenverteilungen hinaus können Sie auch statistische Informationen über das Clusterverhältnis (Cluster ratio) von Indizes, die Anzahl von Blattseiten (Leaf pages) in Indizes, die Anzahl von Tabellenzeilen, die aus ihren ursprünglichen Seiten überlaufen, sowie die Anzahl gefüllter und leerer Seiten in einer Tabelle erfassen. Diese Informationen helfen Ihnen bei der Entscheidung, wann eine Reorganisation von Tabellen und Indizes durchzuführen ist.

Statistische Informationen werden für bestimmte Tabellen und Indizes in der lokalen Datenbank erfasst, wenn Sie das Dienstprogramm RUNSTATS ausführen. Die erfassten Statistikdaten werden in den Systemkatalogtabellen gespeichert.

Statistiken werden nur für die Tabellenpartitionen erfasst, die sich in der Partition befinden, in der Sie das Dienstprogramm ausführen, oder in der ersten Partition der Datenbankpartitionsgruppe, in der die Tabelle enthalten ist.

Anmerkung: Da das Dienstprogramm RUNSTATS keine Verwendung von Kurznamen unterstützt, gehen Sie bei der Aktualisierung von Statistiken für Abfragen in zusammengeschlossenen Datenbanken anders vor. Wenn Abfragen auf eine zusammengeschlossene Datenbank zugreifen, führen Sie RUNSTATS für die Tabellen in allen Datenbanken aus, löschen die Kurznamen, über die auf ferne Tabellen zugegriffen wird, und erstellen diese Kurznamen erneut, um die neuen Statistiken für das Optimierungsprogramm verfügbar zu machen.

Beachten Sie die folgenden Hinweise zur Verbesserung der Effizienz von RUNSTATS und der Nützlichkeit der erfassten Statistikdaten:

- Erfassen Sie Statistikdaten nur für die Spalten, die zur Verknüpfung von Tabellen bzw. in Klauseln WHERE, GROUP BY und ähnlichen von Abfragen verwendet werden. Wenn diese Spalten indexiert sind, können Sie die Spalten mit der Klausel ONLY ON KEY COLUMNS des Befehls RUNSTATS angeben.
- Passen Sie die Werte für die Parameter *num_freqvalues* und *num_quantiles* für bestimmte Tabellen und bestimmte Spalten in Tabellen an.
- Erfassen Sie detaillierte Indexstatistikdaten mit der Klausel SAMPLE DETAILED, um den Umfang der im Hintergrund für detaillierte Indexstatistiken durchgeführten Berechnungen zu verringern. Die Klausel SAMPLE DETAILED verkürzt die Zeit, die zur Erfassung von Statistikdaten benötigt wird, und liefert in den meisten Fällen eine angemessene Genauigkeit.
- Wenn Sie einen Index für eine mit Daten gefüllte Tabelle erstellen, fügen Sie die Klausel COLLECT STATISTICS hinzu, um die Statistikdaten bei der Erstellung des Index zu erfassen.
- Wenn Tabellenzeilen in größerem Umfang hinzugefügt oder gelöscht werden oder wenn Daten in Spalten, für die Statistiken erfasst werden, aktualisiert werden, sollten Sie RUNSTATS erneut ausführen, um die Statistiken zu aktualisieren.
- Da RUNSTATS nur Statistiken in einer einzelnen Partition sammelt, sind die Statistiken weniger genau, wenn die Daten nicht gleichmäßig über alle Partitionen verteilt sind. Wenn Sie eine ungleichmäßige Datenverteilung vermuten, kann es sinnvoll sein, die Daten vor der Ausführung von RUNSTATS mit Hilfe des Befehls REDISTRIBUTE DATABASE PARTITION GROUP erneut auf die Partitionen zu verteilen.

Verteilungsstatistiken werden in folgenden Fällen nicht erfasst:
- Wenn die Konfigurationsparameter *num_freqvalues* und *num_quantiles* auf den Wert null (0) gesetzt sind.
- Wenn die Verteilung von Daten bekannt ist, zum Beispiel wenn jeder Datenwert nur einmal vorkommt.
- Wenn die Spalte einen Datentyp besitzt, für den keine Statistikdaten erfasst werden. Solche Datentypen sind LONG, LOB oder strukturierte Spalten.
- Wenn es sich um Zeilentypen in untergeordneten Tabellen handelt, werden die Statistiken für NPAGES, FPAGES und OVERFLOW der Tabellenebene nicht erfasst.
- Wenn Quantilverteilungsdaten angefordert werden, jedoch nur ein Nichtnullwert in der Spalte vorhanden ist.
- Wenn es sich um erweiterte Indizes oder deklarierte temporäre Tabellen handelt.

Anmerkung: Sie können das Dienstprogramm RUNSTATS für eine deklarierte temporäre Tabelle ausführen. Die Ergebnisstatistikdaten werden jedoch nicht in den Systemkatalogen gespeichert, weil deklarierte Tabellen keine Katalogeinträge besitzen. Allerdings werden die Statistikdaten in Speicherstrukturen abgelegt, welche die Katalog· informationen für deklarierte temporäre Tabellen darstellen. In bestimmten Fällen kann eine Ausführung von RUNSTATS für diese Tabellen daher sinnvoll sein.

Zugehörige Konzepte:
- „Katalogstatistiktabellen" auf Seite 126
- „Statistische Informationen, die erfasst werden" auf Seite 132
- „Katalogstatistiken zu Modellierung und Fallstudien" auf Seite 147
- „Statistiken zur Modellierung von Produktionsdatenbanken" auf Seite 148

- „Allgemeine Regeln zur manuellen Aktualisierung von Katalogstatistiken" auf Seite 151

Zugehörige Tasks:

- „Erfassen von Katalogstatistiken" auf Seite 117

Zugehörige Referenzen:

- „num_freqvalues - Anzahl der häufigsten Werte" auf Seite 506
- „num_quantiles - Anzahl der Quantile für Spalten" auf Seite 507
- „RUNSTATS Command" in *Command Reference*

Erfassen und Analysieren von Systemkatalogstatistiken

Dieser Abschnitt enthält Richtlinien und Anweisungen zur Erfassung von Katalogstatistiken sowie Hinweise zur Analyse der erfassten Daten, um einen besseren Einblick in die Datenverteilung, die Clusterbildung usw. zu erhalten.

Richtlinien für die Erfassung und Aktualisierung von Statistiken

Der Befehl RUNSTATS erfasst Statistiken für Tabellen- und Indexdaten, um genaue Informationen bereitzustellen, die das Optimierungsprogramm zur Auswahl von Zugriffsplänen auswerten kann.

Anmerkung: Das Dienstprogramm RUNSTATS erfasst nur Statistikdaten für Tabellen in der Partition, in der Sie es ausführen. Die RUNSTATS-Ergebnisse aus dieser Partition werden für die anderen Partitionen extrapoliert. Wenn die Datenbankpartition, in der Sie das Dienstprogramm RUNSTATS ausführen, keine Tabellenpartition enthält, wird die Anforderung an die erste Datenbankpartition in der Datenbankpartitionsgruppe gesendet, die eine Partition für die Tabelle enthält.

Verwenden Sie das Dienstprogramm RUNSTATS zur Erfassung von Statistiken in folgenden Situationen:

- Wenn Daten in eine Tabelle geladen und die geeigneten Indizes erstellt wurden.
- Wenn Sie einen neuen Index für eine Tabelle erstellen. Sie müssen das Dienstprogramm RUNSTATS nur für den neuen Index ausführen, wenn die Tabelle seit der letzten Ausführung von RUNSTATS für sie nicht modifiziert wurde.
- Wenn eine Tabelle mit dem Dienstprogramm REORG reorganisiert wurde.
- Wenn die Tabelle und ihre Indizes durch Änderungen, Löschungen und Einfügungen von Daten umfangreich aktualisiert wurden. („Umfangreich" heißt in diesem Fall, dass 10 bis 20 Prozent der Tabellen- und Indexdaten von den Änderungen betroffen sind.)
- Vor dem Binden von Anwendungsprogrammen, deren Leistung von kritischer Bedeutung ist.
- Wenn Sie aktuelle und frühere Statistiken vergleichen wollen. Wenn Sie Statistiken in regelmäßigen Abständen aktualisieren, können Sie Leistungsprobleme frühzeitig erkennen.
- Wenn die Menge der vorab gelesenen Daten (PREFETCHSIZE) geändert wird.
- Wenn das Dienstprogramm REDISTRIBUTE DATABASE PARTITION GROUP zur Umverteilung der Daten verwendet wurde.

Anmerkung: In früheren Versionen von DB2® arbeitete dieser Befehl mit dem Schlüsselwort NODEGROUP anstelle der Schlüsselwörter DATABASE PARTITION GROUP.

Zur Verbesserung der Leistung von RUNSTATS und zur Einsparung von Plattenspeicherplatz für Statistiken sollten Sie in Betracht ziehen, nur die Spalten anzugeben, für die Datenverteilungsstatistiken erfasst werden sollten.

Im Idealfall sollten Anwendungsprogramme nach der Ausführung von RUNSTATS erneut gebunden werden. Das Abfrageoptimierungsprogramm könnte aufgrund der neuen Statistiken einen anderen Zugriffsplan auszuwählen.

Wenn Sie nicht über genügend Zeit verfügen, alle gewünschten Statistiken auf einmal zu erfassen, bietet sich an, RUNSTATS jeweils nur zur Aktualisierung einiger weniger Tabellen und Indizes nach dem Rotationsprinzip auszuführen. Wenn aufgrund der Aktivitäten an den Tabellen zwischen den Zeitpunkten, zu denen Sie RUNSTATS jeweils zu einer Teilaktualisierung ausführen, Inkonsistenzen auftreten, wird während der Abfrageoptimierung eine Warnung (SQL0437W, Ursachencode 6) ausgegeben. Sie verwenden zum Beispiel RUNSTATS zunächst, um Statistikdaten zur Tabellenverteilung zu erfassen. Später führen Sie RUNSTATS aus, um Indexstatistikdaten zu erfassen. Wenn Inkonsistenzen wegen der Aktivitäten an der Tabelle auftreten und während der Abfrageoptimierung erkannt werden, wird die Warnung ausgegeben. Wenn dieser Fall eintritt, sollten Sie RUNSTATS erneut ausführen, um die Verteilungsstatistikdaten zu aktualisieren.

Um sicherzustellen, dass die Indexstatistikdaten mit den Tabellenstatistikdaten synchron sind, sollten Sie RUNSTATS so ausführen, dass Tabellen- und Indexstatistikdaten zu gleicher Zeit erfasst werden. Indexstatistikdaten behalten die Mehrheit der bei der letzten Ausführung von RUNSTATS erfassten Tabellen- und Spaltenstatistikdaten bei. Wenn seit der letzten Erfassung von Tabellenstatistikdaten umfangreiche Änderungen an der Tabelle vorgenommen wurden, geht durch die Erfassung nur der Indexstatistikdaten für diese Tabelle die Synchronisierung der beiden Arten von Statistikdaten auf allen Knoten verloren.

Das Aufrufen des Dienstprogramms RUNSTATS auf einem Produktionssystem kann sich negativ auf die Leistung der Auslastung im Produktionsbetrieb auswirken. Das Dienstprogramm RUNSTATS unterstützt jetzt eine Drosselungsoption, die zur Begrenzung der Leistungsbeeinträchtigung durch die Ausführung von RUNSTATS bei hohem Aufkommen an Datenbankaktivitäten eingesetzt werden kann.

Zugehörige Konzepte:

- „Katalogstatistiken" auf Seite 113
- „Verwendung der Verteilungsstatistiken durch das Optimierungsprogramm" auf Seite 136
- „Automatische Statistikerfassung" auf Seite 124

Zugehörige Tasks:

- „Erfassen von Katalogstatistiken" auf Seite 117
- „Erfassen von Verteilungsstatistiken für bestimmte Spalten" auf Seite 118
- „Erfassen von Indexstatistiken" auf Seite 119

Erfassen von Katalogstatistiken

Die Erfassung von Katalogstatistiken über Tabellen und Indizes dient der Bereitstellung von Informationen, die das Optimierungsprogramm zur Auswahl des besten Zugriffsplans für Abfragen verwendet.

Voraussetzungen:

Sie müssen eine Verbindung zu der Datenbank mit den Tabellen und Indizes herstellen und über eine der folgenden Berechtigungsstufen verfügen:
- SYSADM
- SYSCTRL
- SYSMAINT
- DBADM
- Zugriffsrecht CONTROL für die Tabelle

Vorgehensweise:

Gehen Sie wie folgt vor, um Katalogstatistiken zu erfassen:

1. Stellen Sie eine Verbindung zu der Datenbank her, in der die Tabellen und Indizes enthalten sind, für die Sie die statistischen Informationen erfassen wollen.
2. Führen Sie über die DB2-Befehlszeile den Befehl RUNSTATS mit den gewünschten Optionen aus. Über diese Optionen können Sie die Statistikdaten anpassen, die für Abfragen erfasst werden, die auf Tabellen und Indizes ausgeführt werden.

   Anmerkung: Das Dienstprogramm RUNSTATS erfasst nur Statistikdaten für Tabellen in der Partition, in der Sie es ausführen. Die RUNSTATS-Ergebnisse aus dieser Partition werden für die anderen Partitionen extrapoliert. Wenn die Datenbankpartition, in der Sie das Dienstprogramm RUNSTATS ausführen, keine Tabellenpartition enthält, wird die Anforderung an die erste Datenbankpartition in der Datenbankpartitionsgruppe gesendet, die eine Partition für die Tabelle enthält.
3. Setzen Sie im Anschluss an die Ausführung von RUNSTATS eine Anweisung COMMIT ab, um die Sperren freizugeben.
4. Binden Sie Pakete erneut, die auf Tabellen und Indizes zugreifen, für die Sie Statistiken generiert haben.

Verwenden Sie die Steuerzentrale, wenn Sie eine grafische Benutzerschnittstelle zur Angabe von Optionen und zur Erfassung von Statistiken nutzen wollen.

Zugehörige Konzepte:
- „Katalogstatistiken" auf Seite 113
- „Richtlinien für die Erfassung und Aktualisierung von Statistiken" auf Seite 115

Zugehörige Tasks:
- „Erfassen von Verteilungsstatistiken für bestimmte Spalten" auf Seite 118
- „Erfassen von Indexstatistiken" auf Seite 119
- „Feststellen des Zeitpunkts zur Reorganisation von Tabellen" auf Seite 282

Zugehörige Referenzen:
- „RUNSTATS Command" in *Command Reference*

Erfassen von Verteilungsstatistiken für bestimmte Spalten

Zur Gewährleistung der Effizienz sowohl des Dienstprogramms RUNSTATS als auch der nachfolgenden Abfrageplananalyse ist es sinnvoll, Verteilungsstatistikdaten nur für solche Tabellenspalten zu erfassen, die von Abfragen in Klauseln wie WHERE, GROUP BY und ähnlichen verwendet werden. Darüber hinaus kann es auch nützlich sein, Statistikdaten zur Kardinalität für kombinierte Gruppen von Spalten zu erfassen. Das Optimierungsprogramm verwendet solche Informationen zur Erkennung von Spaltenkorrelationen bei der Abschätzung der Selektivität von Abfragen, die auf die Spalten in der Gruppe verweisen.

Die Beschreibung der folgenden Schritte geht von der Beispielannahme aus, dass die Datenbank **sales** die Tabelle **customers** mit den Indizes **custidx1** und **custidx2** enthält.

Voraussetzungen:

Sie müssen eine Verbindung zu der Datenbank mit den Tabellen und Indizes herstellen und über eine der folgenden Berechtigungsstufen verfügen:
- SYSADM
- SYSCTRL
- SYSMAINT
- DBADM
- Zugriffsrecht CONTROL für die Tabelle

Anmerkung: Das Dienstprogramm RUNSTATS erfasst nur Statistikdaten für Tabellen in der Partition, in der Sie es ausführen. Die RUNSTATS-Ergebnisse aus dieser Partition werden für die anderen Partitionen extrapoliert. Wenn die Datenbankpartition, in der Sie das Dienstprogramm RUNSTATS ausführen, keine Tabellenpartition enthält, wird die Anforderung an die erste Datenbankpartition in der Datenbankpartitionsgruppe gesendet, die eine Partition für die Tabelle enthält.

Vorgehensweise:

Gehen Sie wie folgt vor, um Statistiken für bestimmte Spalten zu erfassen:
1. Stellen Sie eine Verbindung zur Datenbank **sales** her.
2. Führen Sie abhängig von Ihren Anforderungen einen der folgenden Befehle über die DB2-Befehlszeile aus:
   - Zur Erfassung der Verteilungsstatistiken für die Spalten **zip** und **ytdtotal**: :
     ```
     RUNSTATS ON TABLE sales.customers
          WITH DISTRIBUTION ON COLUMNS (zip, ytdtotal)
     ```
   - Zur Erfassung der Verteilungsstatistiken für die gleichen Spalten, jedoch mit Anpassung der Verteilungsstandardwerte: :
     ```
     RUNSTATS ON TABLE sales.customers
          WITH DISTRIBUTION ON
          COLUMNS (zip, ytdtotal NUM_FREQVALUES 50 NUM_QUANTILES 75)
     ```
   - Zur Erfassung der Verteilungsstatistiken für die Spalten, die in **custidx1** und **custidx2** indexiert sind:
     ```
     RUNSTATS ON TABLE sales.customer
              ON KEY COLUMNS
     ```

- Zur Erfassung von Spaltenstatistiken nur für die Spalten **zip** und **ytdtotal** in der Tabelle sowie für eine Spaltengruppe, die die Spalten **region** und **territory** umfasst:

```
RUNSTATS ON TABLE sales.customers
         ON COLUMNS (zip, (region, territory), ytdtotal)
```

Sie können Verteilungsstatistiken auch über die Steuerzentrale erfassen.

Zugehörige Konzepte:

- „Katalogstatistiktabellen" auf Seite 126

Zugehörige Tasks:

- „Erfassen von Katalogstatistiken" auf Seite 117
- „Erfassen von Indexstatistiken" auf Seite 119

Erfassen von Indexstatistiken

Durch die Erfassung von Indexstatistikdaten erhält das Optimierungsprogramm die Möglichkeit, zu beurteilen, ob ein Index zur Erfüllung einer Abfrage verwendet werden sollte.

Die Beschreibung der folgenden Schritte geht von der Beispielannahme aus, dass die Datenbank **sales** die Tabelle **customers** mit den Indizes **custidx1** und **custidx2** enthält.

Voraussetzungen:

Sie müssen eine Verbindung zu der Datenbank mit den Tabellen und Indizes herstellen und über eine der folgenden Berechtigungsstufen verfügen:

- SYSADM
- SYSCTRL
- SYSMAINT
- DBADM
- Zugriffsrecht CONTROL für die Tabelle

Die Ausführung von RUNSTATS mit der Option SAMPLED DETAILED erfordert 2 MB Statistikzwischenspeicher. Ordnen Sie wegen dieses zusätzlichen Speicherbedarfs dem Datenbankkonfigurationsparameter *stat_heap_sz* weitere 488 4-KB-Seiten zu. Wenn der Zwischenspeicher zu klein erscheint, gibt RUNSTATS einen Fehler zurück, bevor das Dienstprogramm versucht, Statistikdaten zu erfassen.

Vorgehensweise:

Gehen Sie wie folgt vor, um detaillierte Statistikdaten für einen Index zu erfassen:

1. Stellen Sie eine Verbindung zur Datenbank **sales** her.
2. Führen Sie abhängig von Ihren Anforderungen einen der folgenden Befehle über die DB2-Befehlszeile aus:
   - Zur Erstellung detaillierter Statistikdaten für die beiden Indizes **custidx1** und **custidx2**:

     ```
     RUNSTATS ON TABLE sales.customers AND DETAILED INDEXES ALL
     ```

- Zur Erstellung detaillierter Statistikdaten für die beiden Indizes, jedoch mit Stichprobenwerten (Sampling) anstelle detaillierter Berechnungen für jeden einzelnen Indexeintrag:

  ```
  RUNSTATS ON TABLE sales.customers AND SAMPLED DETAILED INDEXES ALL
  ```

- Zur Erstellung detaillierter Stichprobenstatistiken für Indizes sowie zur Erstellung von Verteilungsstatistiken für die Tabelle, so dass die Statistikdaten für Indizes und Tabelle konsistent sind:

  ```
  RUNSTATS ON TABLE sales.customers
            WITH DISTRIBUTION ON KEY COLUMNS
            AND SAMPLED DETAILED INDEXES ALL
  ```

Sie können Index- und Tabellenstatistiken auch über die Steuerzentrale erfassen.

Zugehörige Konzepte:

- „Katalogstatistiken" auf Seite 113
- „Katalogstatistiktabellen" auf Seite 126
- „Statistische Informationen, die erfasst werden" auf Seite 132
- „Detaillierte Indexstatistiken" auf Seite 142

Zugehörige Tasks:

- „Erfassen von Katalogstatistiken" auf Seite 117

Erfassen von Statistiken an einer Stichprobe der Tabellendaten

Tabellenstatistiken werden vom Abfrageoptimierungsprogramm bei der Auswahl des besten Zugriffsplans für eine gegebene Abfrage verwendet. Daher spielt es eine wichtige Rolle, dass Statistikdaten aktuell bleiben, um den Status einer Tabelle zu einem beliebigen Zeitpunkt adäquat wiederzugeben. Mit steigendem Aktivitätsvolumen für eine Tabelle sollte auch die Häufigkeit der Statistikerfassung erhöht werden. Mit wachsender Größe von Datenbanken wird es zunehmend wichtiger, effiziente Methoden zur Erfassung von Statistiken zu entwickeln. Zufällige Stichproben, die aus den Tabellendaten erstellt werden und über die Statistiken ermittelt werden, können die Leistung von RUNSTATS verbessern. Für E/A- oder CPU-gebundene Systeme können die Leistungsvorteile enorm sein. Je kleiner die Stichprobe, desto schneller wird RUNSTATS fertig.

Mit Version 8.2 stellt der Befehl RUNSTATS die Option TABLESAMPLE zur Erfassung von Statistiken über eine Stichprobe der Daten in der Tabelle zur Verfügung. Diese Funktion kann die Effizienz der Statistikerfassung erhöhen, da Stichproben nur eine Teilmenge der Daten verwenden. Gleichzeitig stellen Stichprobenmethoden einen hohen Grad an Genauigkeit sicher.

Es gibt zwei Möglichkeiten, die Art und Weise der Stichprobenerstellung anzugeben. Die BERNOULLI-Methode wählt die Stichprobendaten auf der Zeilenebene aus. Bei einer vollständigen Tabellensuche der Datenseite wird jede Zeile für sich geprüft und nach der Wahrscheinlichkeit P, ausgewählt, die durch den numerischen Parameter angegeben ist. Die Statistiken werden nur über diese ausgewählten Zeilen erfasst. In ähnlicher Weise erstellt die Methode SYSTEM eine Stichprobe der Daten auf Seitenebene. Das heißt, jede Seite wird mit einer Wahrscheinlichkeit 1-P/100 ausgewählt oder zurückgewiesen.

Die Leistung für Stichproben auf Seitenebene ist hervorragend, da nur eine E/A-Operation für jede ausgewählte Seite erforderlich ist. Bei Stichproben auf Zeilen-

ebene ist der Ein-/Ausgabeaufwand nicht geringer, da jede Tabellenseite in einer vollständigen Tabellensuche abgerufen wird. Jedoch bieten Stichproben auf Zeilenebene erhebliche Verbesserungen, selbst wenn der Umfang der E/A-Operationen nicht kleiner wird, weil die Erfassung von Statistiken sehr rechenintensiv ist.

Stichproben auf Zeilenebene bieten Stichproben auf Seitenebene gegenüber Vorteile, wenn die Datenwerte eine hohe Clusterbildung aufweisen. Im Vergleich zur Seitenstichprobe vermittelt die Zeilenstichprobe einen genaueren Eindruck von den Daten, da sie P Prozent Zeilen von jeder Datenseite enthält. Bei der Seitenstichprobe befinden jeweils alle Zeilen der P Prozent Seiten in der Stichprobenmenge. Wenn die Zeilen nach dem Zufallsprinzip über die Tabelle verteilt sind, sind die Genauigkeiten von Statistiken über Zeilenstichproben und Seitenstichproben ähnlich.

Jede Stichprobe wird bei jeder Ausführung des Befehls RUNSTATS nach dem Zufallsprinzip neu generiert, sofern nicht die Option REPEATABLE verwendet wird. Mit der Klausel REPEATABLE wird die gleiche Stichprobe generiert wie bei der letzten Ausführung des Befehls RUNSTATS mit der Option TABLESAMPLE. Für Benutzer kann dies in solchen Fällen nützlich sein, in denen eine Generierung konsistenter Statistiken für Tabellen mit konstanten Daten wünschenswert ist.

Zugehörige Konzepte:

- „Stichprobendaten in SQL-Abfragen" auf Seite 96

Zugehörige Referenzen:

- „RUNSTATS Command" in *Command Reference*

Erfassen von Statistiken mit einem Statistikprofil

Das Dienstprogramm RUNSTATS bietet eine Option zur Registrierung und Verwendung eines Statistikprofils, bei dem es sich um eine Gruppe von Optionen handelt, die angeben, welche Statistiken für eine bestimmte Tabelle zu erfassen sind, wie zum Beispiel Tabellenstatistiken, Indexstatistiken oder Verteilungsstatistiken.

Diese Einrichtung vereinfacht die Erfassung von Statistiken, indem es Ihnen die Möglichkeit gibt, die Optionen, die Sie bei der Ausführung des Befehls RUNSTATS angeben, zu speichern, so dass Sie die gleichen Statistiken wiederholt für eine Tabelle erfassen können, ohne die Befehlsoptionen erneut eingeben zu müssen.

Sie können ein Statistikprofil mit und ohne gleichzeitige Erfassung von Statistiken registrieren oder aktualisieren. Um zum Beispiel ein Profil zu registrieren und gleichzeitig Statistiken zu erfassen, führen Sie den Befehl RUNSTATS mit der Option SET PROFILE aus. Soll nur ein Profil registriert werden, ohne tatsächlich Statistiken zu erfassen, führen Sie den Befehl RUNSTATS mit der Option SET PROFILE ONLY aus.

Zur Erfassung von Statistiken unter Verwendung eines bereits registrierten Statistikprofils führen Sie den Befehl RUNSTATS aus, indem Sie nur den Namen der Tabelle und die Option USE PROFILE angeben.

Wenn Sie prüfen wollen, welche Optionen zurzeit in dem Statistikprofil für eine bestimmte Tabelle angegeben sind, können Sie die Katalogtabellen mit der folgenden SELECT-Anweisung abfragen, wobei `tabellenname` der Name der Tabelle ist, für die Sie das Profil abrufen wollen:

```
SELECT STATISTICS_PROFILE FROM SYSIBM.SYSTABLES WHERE NAME = tabellenname
```

Bei der Registrierung eines Benutzerstatistikprofils wird gleichzeitig eine RUNSTATS-Befehlszeichenfolge, die dem Profil entspricht, generiert und in der Spalte STATISTICS_PROFILE der Katalogtabelle SYSIBM.SYSTABLES gespeichert. Wenn die Länge der Befehlszeichenfolge für das registrierte Statistikprofil größer ist als die Spalte STATISTICS_PROFILE, generiert das Dienstprogramm RUNSTATS die Befehlszeichenfolge bis zu der Größe der Spalte und speichert diese abgeschnittene Zeichenfolge in der Spalte SYSTABLES.STATISTICS_PROFILE. Darüber hinaus wird auch eine interne Version des Profils in den Systemkatalogen verwaltet, die nicht abgeschnitten wird.

Automatische Statistikprofilerstellung

Statistikprofile können außerdem automatisch durch die DB2®-Funktion zur automatischen Statistikprofilerstellung generiert werden. Wenn diese Funktion aktiviert ist, werden Informationen zur Datenbankaktivität gesammelt und in einem Abfrage-Feedback-Warehouse gespeichert. Auf der Grundlage dieser Daten wird ein Statistikprofil generiert. Die Aktivierung dieser Funktion kann das Problem der Unsicherheit, welche Statistiken für eine bestimmte Auslastung relevant sind, etwas mildern und die Erfassung einer minimalen Gruppe von Statistiken ermöglichen, um eine optimale Leistung für die Auslastung der Datenbank zu erzielen.

Diese Funktion kann zusammen mit der Funktion zur automatischen Statistikerfassung verwendet werden, die wiederum automatisch Statistikpflegeoperationen auf der Basis der Informationen terminiert, die in dem automatisch generierten Statistikprofil enthalten sind.

Zur Aktivierung dieser Funktion müssen Sie die automatische Tabellenpflege bereits mit Hilfe der entsprechenden Konfigurationsparameter aktiviert haben. Der Konfigurationsparameter AUTO_STATS_PROF aktiviert die Erfassung von Abfragefeedbackdaten, während der Konfigurationsparameter AUTO_PROF_UPD die Generierung eines Statistikprofils zur Verwendung durch die automatische Statistikerfassung aktiviert.

Anmerkung: Die automatische Statistikprofilerstellung kann nur im seriellen DB2-Modus aktiviert werden und wird für Abfragen in zusammengeschlossenen Umgebungen sowie in SMP- oder MPP-Umgebungen blockiert.

Die Statistikprofilgenerierung eignet sich am besten für Umgebungen, in denen große, komplexe Abfragen ausgeführt werden, die viele Vergleichselemente anwenden und häufig Korrelationen in den Daten der Vergleichselementspalten haben und Verknüpfungen und Gruppierungen mehrerer Tabellen erfordern. Sie eignet sich weniger für Umgebungen mit einer in erster Linie transaktionsorientierten Auslastung.

Es gibt zwei unterschiedliche Möglichkeiten, diese Funktion zu verwenden:

- In einer Testumgebung. Setzen Sie AUTO_STATS_PROF und AUTO_PROF_UPD auf ON in Testsystemen, in denen der Leistungsaufwand der Laufzeitüberwachung leicht toleriert werden kann. Wenn das Testsystem mit realistischen Daten und Abfragen arbeitet, ermöglicht dies eine Ermittlung der tatsächlichen Korrelationen und Einstellungen der Statistikparameter für RUNSTATS, die wiederum im Statistikprofil gespeichert werden. Diese Profile können anschließend auf das Produktionssystem übertragen werden, auf dem die Abfragen von ihnen profitieren können, ohne den Überwachungsaufwand zu verursachen.
- Zur Untersuchung von Leistungsproblemen für bestimmte Abfragen in einer Produktionsumgebung. Wenn Leistungsprobleme für eine bestimmte Gruppe

von Abfragen festgestellt werden, die fehlerhaften Statistiken oder Korrelationen zugeschrieben werden können, können Sie AUTO_STATS_PROF aktivieren und die Zielauslastung über einen Zeitraum ausführen. Die automatische Statistikprofilgenerierung analysiert das Abfragefeedback und erstellt Empfehlungen in den Tabellen SYSTOOLS.OPT_FEEDBACK_RANKING*. Sie können diese Empfehlungen einsehen und die Statistikprofile manuell auf der Grundlage der Empfehlungen verfeinern. Wenn DB2 die Statistikprofile auf der Grundlage dieser Empfehlungen automatisch aktualisieren soll, aktivieren Sie AUTO_PROF_UPD, wenn Sie AUTO_STATS_PROF aktivieren.

Anmerkung: Die Überwachung der Abfragen und das Speichern von Abfragefeedbackdaten im Feedback-Warehouse ist mit einigem Leistungsaufwand verbunden.

Erstellen des Warehouse für Abfragefeedback: Das Feedback-Warehouse besteht aus drei Tabellen im Schema SYSTOOLS, in denen Informationen zu den Vergleichselementen gespeichert werden, die bei der Ausführung von Abfragen angetroffen werden. Diese drei Tabellen heißen OPT_FEEDBACK_QUERY, OPT-_FEEDBACK_PREDICATE und OPT_FEEDBACK_PREDICATE_COLUMN.

Zur Verwendung der automatischen Statistikprofilgenerierung müssen Sie zunächst das Abfrage-Feedback-Warehouse mit Hilfe der gespeicherten Prozedur SYSINSTALLOBJECTS erstellen. Diese gespeicherte Prozedur ist die allgemeine gespeicherte Prozedur zum Erstellen und Löschen von Objekten im Schema SYSTOOLS.

Rufen Sie die gespeicherte Prozedur SYSINSTALLOBJECTS wie folgt auf:

```
call SYSINSTALLOBJECTS ( toolname, aktion, tabellenbereichsname, schemaname)
```

Dabei gilt Folgendes:

toolname
Gibt den Namen des Tools an, dessen Objekte zu erstellen oder löschen sind. In diesem Fall ist dies "ASP" oder "AUTO STATS PROFILING".

aktion Gibt die auszuführende Aktion an: 'C' für Erstellen (Create) und 'D' für Löschen (Drop).

tabellenbereichsname
Der Name des Tabellenbereichs, in dem die Tabellen für das Feedback-Warehouse erstellt werden sollen. Dieser Eingabeparameter ist optional. Wenn er nicht angegeben wird, wird der Standardbenutzertabellenbereich verwendet.

schemaname
Der Name des Schemas, in dem die Objekte erstellt oder gelöscht werden. Dieser Parameter wird momentan nicht verwendet.

Wenn Sie zum Beispiel das Feedback-Warehouse in Tabellenbereich "A" erstellen wollen, geben Sie Folgendes ein: `call SYSINSTALLOBJECTS ("ASP", 'C', "A", "")`

Zugehörige Konzepte:

- „Automatische Statistikerfassung" auf Seite 124

Zugehörige Tasks:

- „Verwenden der automatischen Statistikerfassung" auf Seite 125

Automatische Statistikerfassung

Das DB2®-Optimierungsprogramm verwendet Katalogstatistiken, um den effizientesten Zugriffsplan für eine gegebene Abfrage zu ermitteln. Wenn die Statistiken für eine Tabelle oder einen Index nicht auf dem neuesten Stand oder unvollständig sind, könnte das Optimierungsprogramm einen Plan auswählen, der nicht optimal ist, so dass die Abfrageausführung verlangsamt würde. Allerdings ist die Entscheidung, welche Statistiken für eine gegebene Auslastung zu erfassen sind, recht komplex und die Pflege aktueller Statistiken zeitaufwendig.

Mit der automatischen Statistikerfassung, die Teil der Funktion zur automatischen Tabellenverwaltung von DB2 ist, können Sie DB2 die Entscheidung überlassen, welche Statistiken für Ihre Auslastung erforderlich sind und welche Statistiken aktualisiert werden müssen. Bei aktivierter automatischer Statistikerfassung führt DB2 das Dienstprogramm RUNSTATS automatisch im Hintergrund aus, um sicherzustellen, dass die korrekten Statistiken erfasst und gepflegt werden.

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

- Die Statistikerfassung erfolgt durch eine gedrosselte Ausführung von RUNSTATS. Die Drosselung begrenzt je nach aktueller Datenbankaktivität die Menge der Ressourcen, die vom Dienstprogramm RUNSTATS beansprucht werden: wenn die Datenbankaktivitäten zunehmen, läuft das Dienstprogramm RUNSTATS langsamer, so dass es weniger Ressourcen benötigt.
- Es wird nur eine minimale Gruppe von Statistiken zur Optimierung der Leistung erfasst. Dies wird durch die Verwendung von Statistikprofilen erreicht, bei der Informationen über vergangene Datenbankaktivitäten genutzt werden, um zu bestimmen, welche Statistiken für die Datenbankauslastung erforderlich sind und wie schnell diese Statistiken in Anbetracht der Arten von Aktivitäten in der Datenbank veralten.
- Nur Tabellen mit hohen Graden an Aktivität (gemessen an der Anzahl der Aktualisierungen, Löschungen und Einfügungen) werden für die Statistikerfassung in Betracht gezogen. Für große Tabellen (die aus mehr als 4000 Seiten bestehen) werden außerdem Stichproben erstellt, um festzustellen, ob die Statistiken durch die intensive Tabellenaktivität tatsächlich beeinflusst wurden. Statistiken für solch große Tabellen werden nur erfasst, wenn dies gerechtfertigt ist.
- Das Dienstprogramm RUNSTATS wird automatisch zur Ausführung während des optimalen Verwaltungsfensters terminiert, das in der Definition 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.
- Während der Ausführung der automatischen Statistikerfassung bleiben die betroffenen Tabellen weiterhin für reguläre Datenbankaktivitäten (Aktualisierungen, Einfügungen, Löschungen) verfügbar, so als ob RUNSTATS nicht für die Tabelle ausgeführt würde.

Zugehörige Konzepte:

- „Erfassen von Statistiken mit einem Statistikprofil" auf Seite 121

Zugehörige Tasks:

- „Verwenden der automatischen Statistikerfassung" auf Seite 125

Zugehörige Referenzen:

- „util_impact_lim - Richtlinie für Exemplarauslastungswirkung" auf Seite 536
- „autonomic_switches - Schalter für automatische Verwaltung" auf Seite 508

Verwenden 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. In einer seriellen Umgebung können Sie diese Funktionalität optional noch erweitern, indem Sie Abfragedaten sammeln und Statistikprofile generieren, die DB2 bei der automatischen Erfassung der exakten Gruppe der für Ihre Auslastung erforderlichen Statistiken unterstützen. Diese Option steht in SMP- und MPP-Umgebungen sowie in zusammengeschlossenen Umgebungen nicht zur Verfügung.

Vorgehensweise:

Sie können diese Funktion entweder über die Tools der grafischen Benutzerschnittstelle oder über die Befehlszeilenschnittstelle aktivieren.

- Gehen Sie wie folgt vor, um Ihre Datenbank für die automatische Statistikerfassung über die Tools der grafischen Benutzerschnittstelle zu konfigurieren:
  1. Öffnen Sie den Assistenten „Automatische Verwaltung konfigurieren" entweder über die Steuerzentrale, indem Sie ein Datenbankobjekt mit der rechten Maustaste anklicken, oder über die Diagnosezentrale, indem Sie das Datenbankexemplar, das Sie zur automatischen Statistikerfassung konfigurieren wollen, mit der rechten Maustaste anklicken. Wählen Sie **Automatische Verwaltung konfigurieren** im Kontextmenü aus.
  2. In diesem Assistenten können Sie die automatische Statistikerfassung aktivieren, die Tabellen angeben, aus denen die Statistiken automatisch zu erfassen sind, und ein Verwaltungsfenster für die Ausführung des Dienstprogramms RUNSTATS definieren.
  3. OPTIONAL: Zur Aktivierung der automatischen Statistikprofilerstellung setzen Sie die folgenden beiden Konfigurationsparameter über die Befehlszeilenschnittstelle auf ON:
     - AUTO_STATS_PROF
     - AUTO_PROF_UPD
- Gehen Sie wie folgt vor, um Ihre Datenbank für die automatische Statistikerfassung über die Befehlszeilenschnittstelle zu konfigurieren:
  1. Setzen Sie jeden der folgenden Konfigurationsparameter auf den Wert ON:
     - AUTO_MAINT
     - AUTO_TBL_MAINT
     - AUTO_RUNSTATS
  2. OPTIONAL: Zur Aktivierung der automatischen Statistikprofilerstellung setzen Sie die folgenden beiden Konfigurationsparameter auf ON:
     - AUTO_STATS_PROF
     - AUTO_PROF_UPD

Zugehörige Konzepte:

- „Erfassen von Statistiken mit einem Statistikprofil" auf Seite 121
- „Automatische Statistikerfassung" auf Seite 124

Zugehörige Referenzen:

- „autonomic_switches - Schalter für automatische Verwaltung" auf Seite 508

Erfasste Statistiken

Dieser Abschnitt listet die Katalogstatistiktabellen auf und beschreibt die Verwendung der Felder in diesen Tabellen. Die Abschnitte, die sich an die Beschreibungen der Statistiktabellen anschließen, erläutern die Art von Daten, die in den Tabellen erfasst und gespeichert werden kann.

Katalogstatistiktabellen

Die folgenden Tabellen bieten Informationen zu den Systemkatalogtabellen, die Katalogstatistiken enthalten, und zeigen die RUNSTATS-Optionen, die zur Erfassung bestimmter Statistiken verwendet werden.

Tabelle 18. Tabellenstatistiken (SYSCAT.TABLES und SYSSTAT.TABLES)

Statistik	Beschreibung	RUNSTATS-Option	
		Tabelle	**Indizes**
FPAGES	Anzahl der von einer Tabelle verwendeten Seiten	Ja	Ja
NPAGES	Anzahl der Zeilen enthaltenden Seiten	Ja	Ja
OVERFLOW	Anzahl der Überlaufzeilen	Ja	Nein
CARD	Anzahl der Zeilen in der Tabelle (Kardinalität)	Ja	Ja (Anm. 1)
ACTIVE_BLOCKS	Für MDC-Tabellen: die Gesamtzahl belegter Blöcke	Ja	Nein

Anmerkung:
1. Wenn für die Tabelle keine Indizes erstellt wurden und Sie die Indexstatistik anfordern, wird die CARD-Statistik mit keinem neuen Wert aktualisiert. Die vorige CARD-Statistik wird beibehalten.

Tabelle 19. Spaltenstatistiken (SYSCAT.COLUMNS und SYSSTAT.COLUMNS)

Statistik	Beschreibung	RUNSTATS-Option	
		Tabelle	**Indizes**
COLCARD	Anzahl der unterschiedlichen Werte der Spalte (Spaltenkardinalität)	Ja	Ja (Anm. 1)
AVGCOLLEN	Durchschnittslänge der Spalte	Ja	Ja (Anm. 1)
HIGH2KEY	Zweithöchster Wert der Spalte	Ja	Ja (Anm. 1)
LOW2KEY	Zweitniedrigster Wert der Spalte	Ja	Ja (Anm. 1)
NUMNULLS	Die Anzahl Nullwerte (NULLs) in einer Spalte	Ja	Ja (Anm. 1)
SUB_COUNT	Die durchschnittliche Anzahl von Unterelementen	Ja	Nein (Anm. 2)
SUB_DELIM_LENGTH	Durchschnittslänge jedes Begrenzers, der ein Unterelement trennt	Ja	Nein (Anm. 2)

Tabelle 19. Spaltenstatistiken (SYSCAT.COLUMNS und SYSSTAT.COLUMNS) (Forts.)

Statistik	Beschreibung	RUNSTATS-Option	
		Tabelle	**Indizes**
Anmerkung: 1. Spaltenstatistiken werden für die erste Spalte im Indexschlüssel erfasst. 2. Diese Statistiken stellen Informationen zu Daten in Spalten bereit, die eine Reihe von Unterfeldern bzw. Unterelementen enthalten, die durch Leerzeichen begrenzt werden. Die Werte SUB_COUNT und SUB_DELIM_LENGTH werden nur für Einzelbyte-Zeichenfolgespalten der Typen CHAR, VARCHAR, GRAPHIC und VARGRAPHIC erfasst.			

Tabelle 20. Spaltengruppenstatistiken (SYSCAT.COLGROUPS und SYSSTAT.COLGROUPS)

Statistik	Beschreibung	RUNSTATS-Option	
		Tabelle	**Indizes**
COLGROUPCARD	Kardinalität der Spaltengruppe	Ja	Nein

Anmerkung: Die Verteilungsstatistiken für Spaltengruppen in den folgenden beiden Tabellen werden nicht von RUNSTATS erfasst. Sie können diese Statistiken jedoch manuell aktualisieren.

Tabelle 21. Verteilungsstatistiken für Spaltengruppen (SYSCAT.COLGROUPDIST und SYSSTAT.COLGROUPDIST)

Statistik	Beschreibung	RUNSTATS-Option	
		Tabelle	**Indizes**
TYPE	F = Häufigkeitswert Q = Quantilwert	Ja	Nein
ORDINAL	Ordinalzahl der Spalte in der Gruppe	Ja	Nein
SEQNO	Folgenummer *n*, die den *n*-ten TYPE-Wert darstellt	Ja	Nein
COLVALUE	Der Datenwert als Zeichenliteral oder Nullwert	Ja	Nein

Tabelle 22. Verteilungsstatistiken für Spaltengruppen 2 (SYSCAT.COLGROUPDISTCOUNTS und SYSSTAT.COLGROUPDISTCOUNTS)

Statistik	Beschreibung	RUNSTATS-Option	
		Tabelle	**Indizes**
TYPE	F = Häufigkeitswert Q = Quantilwert	Ja	Nein
SEQNO	Folgenummer *n*, die den *n*-ten TYPE-Wert darstellt	Ja	Nein

Tabelle 22. Verteilungsstatistiken für Spaltengruppen 2 (SYSCAT.COLGROUPDISTCOUNTS und SYSSTAT.COLGROUPDISTCOUNTS) (Forts.)

Statistik	Beschreibung	RUNSTATS-Option	
		Tabelle	**Indizes**
VALCOUNT	Bei TYPE = F ist VALCOUNT die Anzahl Vorkommen von COLVALUE-Werten für die Spaltengruppe, die durch diese Folgenummer (SEQNO) angegeben ist. Bei TYPE = Q ist VALCOUNT die Anzahl Zeilen, deren Wert kleiner oder gleich COLVALUE-Werten für die Spaltengruppe mit dieser Folgenummer ist.	Ja	Nein
DISTCOUNT	Bei TYPE = Q enthält diese Spalte die Anzahl unterschiedlicher Werte, die kleiner oder gleich COLVALUE-Werten für die Spaltengruppe mit dieser Folgenummer (SEQNO) sind. Null, falls nicht verfügbar.	Ja	Nein

Tabelle 23. Indexstatistiken (SYSCAT.INDEXES und SYSSTAT.INDEXES)

Statistik	Beschreibung	RUNSTATS-Option	
		Tabelle	**Indizes**
NLEAF	Anzahl der Indexblattseiten (leaf pages)	Nein	Ja
NLEVELS	Anzahl der Indexstufen	Nein	Ja
CLUSTERRATIO	Grad der Clusterbildung der Tabellendaten	Nein	Ja (Anm. 2)
CLUSTERFACTOR	Feinerer Grad der Clusterbildung	Nein	Detailliert (Anm. 1,2)
DENSITY	Verhältnis (Prozentsatz) von SEQUENTIAL_ PAGES zur Anzahl der Seiten im Bereich der vom Index belegten Seiten (Anm. 3)	Nein	Ja
FIRSTKEYCARD	Anzahl der unterschiedlichen Werte in der ersten Spalte des Index	Nein	Ja

Tabelle 23. Indexstatistiken (SYSCAT.INDEXES und SYSSTAT.INDEXES) (Forts.)

Statistik	Beschreibung	RUNSTATS-Option	
		Tabelle	Indizes
FIRST2KEYCARD	Anzahl der unterschiedlichen Werte in den ersten beiden Spalten des Index	Nein	Ja
FIRST3KEYCARD	Anzahl der unterschiedlichen Werte in den ersten drei Spalten des Index	Nein	Ja
FIRST4KEYCARD	Anzahl der unterschiedlichen Werte in den ersten vier Spalten des Index	Nein	Ja
FULLKEYCARD	Anzahl der unterschiedlichen Werte in allen Spalten des Index, mit Ausnahme der Schlüsselwerte in einem Index des Typs 2, für die alle Satz-IDs (RIDs) als gelöscht markiert sind	Nein	Ja
PAGE_FETCH_PAIRS	Geschätzte Anzahl der Seitenabrufe für verschiedene Puffergrößen	Nein	Detailliert (Anm. 1,2)
SEQUENTIAL_PAGES	Anzahl der Blattseiten (äußersten Seiten), die auf der Platte in der durch den Indexschlüssel definierten Reihenfolge mit wenigen oder keinen großen dazwischen liegenden Lücken gespeichert sind	Nein	Ja
AVERAGE_SEQUENCE_PAGES	Durchschnittliche Anzahl von Indexseiten, auf die in Reihenfolge zugegriffen werden kann. Dies ist die Anzahl von Indexseiten, die von Vorablesefunktionen als in der richtigen Reihenfolge vorliegend erkannt werden können.	Nein	Ja

Tabelle 23. Indexstatistiken (SYSCAT.INDEXES und SYSSTAT.INDEXES) (Forts.)

Statistik	Beschreibung	RUNSTATS-Option	
		Tabelle	**Indizes**
AVERAGE_RANDOM_PAGES	Durchschnittliche Anzahl von Indexseiten, auf die zwischen den sequenziellen Seitenzugriffen wahlfrei zugegriffen werden muss	Nein	Ja
AVERAGE_SEQUENCE_GAP	Lücke zwischen den Seitenfolgen	Nein	Ja
AVERAGE_SEQUENCE_FETCH_PAGES	Durchschnittliche Anzahl von Tabellenseiten, auf die in Reihenfolge zugegriffen werden kann. Dies ist die Anzahl von Tabellenseiten, die von Vorablesefunktionen als in der richtigen Reihenfolge vorliegend erkannt werden können, wenn sie versuchen, Tabellenzeilen über den Index abzurufen.	Nein	Ja (Anm. 4)
AVERAGE_RANDOM_FETCH_PAGES	Durchschnittliche Anzahl von Tabellenseiten, auf die ein wahlfreier Zugriff erfolgt, zwischen sequenziellen Seitenzugriffen beim Abrufen von Tabellenzeilen über den Index	Nein	Ja (Anm. 4)
AVERAGE_SEQUENCE_FETCH_GAP	Lücke zwischen sequenziellen Zugriffen beim Abrufen von Tabellenzeilen über den Index	Nein	Ja (Anm. 4)
NUMRIDS	Anzahl von Satz-IDs (RIDs) im Index, einschließlich gelöschter Satz-IDs in Indizes des Typs 2	Nein	Ja
NUMRIDS_DELETED	Gesamtanzahl von Satz-IDs (RIDs), die im Index als gelöscht markiert sind, außer den Satz-IDs in Blattseiten, in denen alle Satz-IDs als gelöscht markiert sind	Nein	Ja

Tabelle 23. Indexstatistiken (SYSCAT.INDEXES und SYSSTAT.INDEXES) (Forts.)

Statistik	Beschreibung	RUNSTATS-Option	
		Tabelle	**Indizes**
NUM_EMPTY_LEAFS	Gesamtanzahl von Blattseiten, in denen alle Satz-IDs als gelöscht markiert sind	Nein	Ja

Anmerkung:

1. Detaillierte Indexstatistikdaten werden erfasst, indem die Klausel DETAILED im Befehl RUNSTATS angegeben wird.
2. CLUSTERFACTOR und PAGE_FETCH_PAIRS werden mit der Klausel DETAILED nur dann erfasst, wenn die Tabelle eine beträchtliche Größe aufweist. Wenn die Tabelle größer als ca. 25 Seiten ist, werden die Statistikdaten für die Spalten CLUSTERFACTOR und PAGE_FETCH_PAIRS gesammelt. In diesem Fall ist CLUSTERRATIO (Clusterverhältnis) -1 (d. h. wird nicht gesammelt). Wenn die Tabelle relativ klein ist, wird nur die Spalte CLUSTERRATIO vom Dienstprogramm RUNSTATS ausgefüllt, während die Spalten CLUSTERFACTOR und PAGE_FETCH_PAIRS nicht berechnet werden. Wenn die Klausel DETAILED nicht angegeben wird, werden nur die Statistikdaten für CLUSTERRATIO erfasst.
3. Diese Statistik ermittelt den Prozentsatz von Seiten in der Datei, die den Index enthalten, der zu dieser Tabelle gehört. Für eine Tabelle, die nur einen definierten Index hat, sollte DENSITY normalerweise gleich 100 sein. DENSITY wird vom Optimierungsprogramm zur Abschätzung verwendet, wie viele irrelevante Seiten von anderen Indizes durchschnittlich vielleicht gelesen werden, wenn die Indexseiten vorabgelesen würden.
4. Diese Statistiken können nicht berechnet werden, wenn sich die Tabelle in einem DMS-Tabellenbereich befindet.
5. Statistiken über Vorablesezugriffe werden beim Laden von Indizes (LOAD INDEX) bzw. beim Erstellen von Indizes (CREATE INDEX) nicht erfasst, und zwar auch dann nicht, wenn die Funktion zum Erfassen von Statistiken beim Aufrufen des Befehls angegeben wird. Statistiken über Vorablesezugriffe werden ebenfalls nicht erfasst, wenn der Konfigurationsparameter für die Markierung der Sequenzerkennung (seqdetect) ausgeschaltet ist.

Tabelle 24. Spaltenverteilungsstatistiken (SYSCAT.COLDIST und SYSSTAT.COLDIST)

Statistik	Beschreibung	RUNSTATS-Option	
		Tabelle	**Indizes**
DISTCOUNT	Wenn TYPE den Wert Q hat, die Anzahl unterschiedlicher Werte, die kleiner oder gleich dem Statistikwert COLVALUE sind	Verteilung (Anm. 2)	Nein
TYPE	Gibt an, ob die Zeile statistische Daten über die Häufigkeit der Werte oder über Quantilwerte enthält	Verteilung	Nein
SEQNO	Die Stelle in der Häufigkeitsrangfolge einer Folgenummer, die als Hilfe zur eindeutigen Bestimmung der Zeile in der Tabelle verwendet werden kann	Verteilung	Nein
COLVALUE	Datenwert, für den Statistikdaten zur Häufigkeit oder zu Quantilwerten erfasst werden	Verteilung	Nein

Tabelle 24. Spaltenverteilungsstatistiken (SYSCAT.COLDIST und SYSSTAT.COLDIST) (Forts.)

Statistik	Beschreibung	RUNSTATS-Option	
		Tabelle	**Indizes**
VALCOUNT	Häufigkeit, mit der der Datenwert in der Spalte auftritt, oder bei Quantilwerten die Anzahl der Werte, die kleiner oder gleich dem Datenwert (COLVALUE) sind	Verteilung	Nein

Anmerkung:
1. Verteilungsstatistikdaten über Spalten werden gesammelt, indem die Klausel WITH DISTRIBUTION im Befehl RUNSTATS angegeben wird. Beachten Sie, dass Verteilungsstatistikdaten **nicht** gesammelt werden, wenn das Ausmaß der Ungleichmäßigkeit der Werteverteilung in den Spaltenwerten nicht hoch genug ist.
2. DISTCOUNT wird nur für Spalten erfasst, die die erste Schlüsselspalte in einem Index bilden.

Zugehörige Konzepte:

- „Katalogstatistiken" auf Seite 113
- „Statistische Informationen, die erfasst werden" auf Seite 132
- „Statistiken für benutzerdefinierte Funktionen" auf Seite 145
- „Statistiken zur Modellierung von Produktionsdatenbanken" auf Seite 148

Statistische Informationen, die erfasst werden

Wenn Sie das Dienstprogramm RUNSTATS für eine Tabelle oder für eine Tabelle mit zugehörigen Indizes ausführen, werden immer die folgenden Arten von Statistikinformationen in den Systemkatalogtabellen gespeichert:

Für eine Tabelle und einen Index:
- Die Anzahl der verwendeten Seiten
- Die Anzahl der Seiten, die Zeilen enthalten
- Die Anzahl der Zeilen, die überlaufen
- Die Anzahl von Zeilen in der Tabelle (Kardinalität)
- Für MDC-Tabellen: die Anzahl von Blöcken, die Daten enthalten

Für jede Spalte in der Tabelle und die erste Spalte im Indexschlüssel:
- Die Kardinalität der Spalte
- Die Durchschnittslänge der Spalte
- Der zweithöchste Wert in der Spalte
- Der zweitniedrigste Wert in der Spalte
- Die Anzahl von Nullwerten (NULL) in der Spalte

Für Gruppen von Spalten, die Sie angeben:
- Ein auf einer Zeitmarke basierender Name für die Spaltengruppe
- Die Kardinalität der Spaltengruppe

Nur für Indizes:
- Die Anzahl von Blattseiten (leaf pages)
- Die Anzahl von Indexstufen
- Der Grad der Clusterbildung der Tabellendaten in Bezug auf den Index
- Das Verhältnis von Blattseiten auf dem Datenträger in der Reihenfolge des Indexschlüssels zur Anzahl von Seiten in dem Bereich von Seiten, die durch den Index belegt werden
- Die Anzahl unterschiedlicher Werte in der Spalte des Index

- Die Anzahl unterschiedlicher Werte in den ersten zwei, drei und vier Spalten des Index
- Die Anzahl unterschiedlicher Werte in allen Spalten des Index
- Die Anzahl von Blattseiten, die sich auf dem Datenträger in der Reihenfolge des Indexschlüssels mit wenigen oder keinen dazwischen liegenden Lücken befinden
- Die Anzahl von Seiten, in denen alle Satz-IDs (RIDs) als gelöscht markiert sind
- Die Anzahl als gelöscht markierter Satz-IDs in Seiten, in denen nicht alle Satz-IDs als gelöscht markiert sind

Wenn Sie detaillierte Statistiken für einen Index anfordern, speichern Sie auch feinere Informationen über den Grad der Clusterbildung der Tabelle in Bezug auf den Index sowie die Seitenabrufschätzwerte für verschiedene Puffergrößen.

Darüber hinaus können Sie die folgenden Arten von Statistiken über Tabellen und Indizes erfassen:

- Statistiken zur Datenverteilung

  Das Optimierungsprogramm nutzt die Statistiken über die Datenverteilung zur Abschätzung effizienter Zugriffspläne für Tabellen, in denen Datenwerte nicht gleichmäßig verteilt sind und Spalten eine beträchtliche Anzahl mehrfach auftretender Werte aufweisen.
- Detaillierte Statistiken zu Indizes

  Das Optimierungsprogramm nutzt detaillierte Indexstatistiken zur Ermittlung, wie effizient der Zugriff auf eine Tabelle über einen Index realisiert werden kann.
- Statistiken zu Unterelementen

  Das Optimierungsprogramm nutzt Unterelementstatistiken für LIKE-Vergleichselemente, insbesondere für solche, die nach einem in eine Zeichenfolge eingebetteten Muster suchen, wie zum Beispiel `LIKE %disk%`.

Zugehörige Konzepte:

- „Katalogstatistiken" auf Seite 113
- „Katalogstatistiktabellen" auf Seite 126

Zugehörige Tasks:

- „Erfassen von Katalogstatistiken" auf Seite 117
- „Erfassen von Verteilungsstatistiken für bestimmte Spalten" auf Seite 118
- „Erfassen von Indexstatistiken" auf Seite 119

Verteilungsstatistiken

Sie können zwei Arten von Statistiken über die Datenverteilung erfassen:

- Statistiken zur Häufigkeit von Datenwerten

  Diese Statistiken stellen Informationen über die Spalte und den Datenwert mit der höchsten Anzahl mehrfacher Vorkommen, die nächsthöchste Anzahl mehrfach vorkommender Werte usw. bis zu der Stufe, die durch den Wert des Datenbankkonfigurationsparameters *num_freqvalues* angegeben wird, bereit. Um die Erfassung der Statistik über die Häufigkeit von Werten zu inaktivieren, müssen Sie den Parameter *num_freqvalues* auf den Wert 0 setzen.

  Sie können den Wert für *num_freqvalues* auch als RUNSTATS-Optionen für jede Tabelle und jede bestimmte Spalte festlegen.
- Quantilstatistiken

  Diese Statistiken liefern Informationen darüber, wie Datenwerte in Relation zu anderen Werten verteilt sind. Die Statistiken der so genannten K-Quantile stellen

den Wert V dar, bei oder unter dem mindestens K Werte liegen. Ein K-Quantil lässt sich durch Sortieren der Werte in aufsteigender Reihenfolge ermitteln. Der K-Quantilwert ist der Wert an der K-ten Position vom unteren Ende des Wertebereichs betrachtet.

Um die Anzahl der Abschnitte anzugeben, in die die Spaltendatenwerte gruppiert werden sollen, setzen Sie den Datenbankkonfigurationsparameter *num_quantiles* auf einen Wert zwischen 2 und 32.767. Der Standardwert ist 20. Dieser Wert stellt einen Schätzfehler durch das Optimierungsprogramm von maximal plus/minus 2,5% für ein beliebiges Vergleichselement mit einem der Operatoren „=", „>" oder „<" sowie einen maximalen Fehler von plus/minus 5% für ein beliebiges BETWEEN-Vergleichselement sicher. Zur Inaktivierung der Erfassung der Quantilstatistiken setzen Sie den Parameter *num_quantiles* auf den Wert 0 oder 1.

Sie können den Parameter *num_quantiles* für jede Tabelle und für bestimmte Spalten definieren.

Anmerkung: Wenn Sie höhere Werte für die Parameter *num_freqvalues* und *num_quantiles* angeben, werden bei der Ausführung von RUNSTATS mehr CPU-Kapazitäten und Speicher entsprechend dem Datenbankkonfigurationsparameter *stat_heap_sz* benötigt.

Wann sind Verteilungsstatistiken zu erfassen

Bei der Entscheidung, ob Verteilungsstatistikdaten für eine bestimmte Tabelle erstellt und aktualisiert werden sollten, spielen zwei Faktoren eine ausschlaggebende Rolle:

- Ob Anwendungen mit statischem oder dynamischem SQL arbeiten.

  Verteilungsstatistiken sind am besten für dynamisches SQL sowie für SQL, das keine Hostvariablen verwendet, geeignet. Wenn SQL mit Hostvariablen verwendet wird, nutzt das Optimierungsprogramm die Verteilungsstatistikdaten nur in begrenztem Umfang.
- Ob die Datenwerte in Spalten gleichmäßig verteilt sind.

  Erstellen Verteilungsstatistiken, wenn mindestens in einer Spalte der Tabelle die Datenwerte höchst *ungleichmäßig* verteilt sind und diese Spalte häufig in Gleichheits- bzw. Bereichsvergleichselementen auftritt, wie zum Beispiel in folgenden Klauseln:

```
WHERE C1 = KEY;
WHERE C1 IN (KEY1, KEY2, KEY3);
WHERE (C1 = KEY1) OR (C1 = KEY2) OR (C1 = KEY3);
WHERE C1 <= KEY;
WHERE C1 BETWEEN KEY1 AND KEY2;
```

Zwei Arten ungleichmäßiger Datenverteilungen treten möglicherweise zusammen auf:

- Datenwerte können in einem oder mehreren Unterintervallen gehäuft auftreten, anstatt gleichmäßig zwischen dem höchsten und niedrigsten Datenwert verteilt zu sein. Betrachten Sie die folgende Spalte, in der die Datenwerte im Bereich (5,10) eine Häufung aufweisen:

C1
0,0
5,1
6,3
7,1
8,2

C1
8,4
8,5
9,1
93,6
100,0

Quantilstatistiken helfen dem Optimierungsprogramms bei der Einschätzung dieser Art von Datenverteilung.

Zur Ermittlung, ob Spaltendaten nicht gleichmäßig verteilt sind, können Sie eine Abfrage wie in folgendem Beispiel ausführen:

```
SELECT C1, COUNT(*) AS OCCURRENCES
  FROM T1
GROUP BY C1
ORDER BY OCCURRENCES DESC;
```

- Mehrfach auftretende Datenwerte können häufig vorkommen. Betrachten Sie eine Spalte, in der die Daten mit den folgenden Häufigkeiten verteilt sind:

Datenwert	Häufigkeit
20	5
30	10
40	10
50	25
60	25
70	20
80	5

Zur Unterstützung des Optimierungsprogramms bei der Einschätzung mehrfach auftretender Werte sollten Sie sowohl Quantilstatistiken als auch Statistiken zur Wertehäufigkeit erstellen.

Wann sind nur Indexstatistiken zu erfassen

Sie können Statistiken nur für Indexdaten in folgenden Situationen erfassen:
- Seit der letzten Ausführung des Dienstprogramms RUNSTATS wurde ein neuer Index erstellt, und Sie möchten nicht erneut statistische Daten für die Tabellendaten sammeln.
- Es wurden viele Änderungen an den Daten vorgenommen, die die erste Spalte eines Index betreffen.

Welche Stufe an statistischer Genauigkeit ist anzugeben

Zur Festlegung der Genauigkeit, mit der Verteilungsstatistiken erfasst werden, definieren Sie die Datenbankkonfigurationsparameter *num_quantiles* und *num_freqvalues* mit den entsprechenden Werten. Sie können diese Parameter auch als RUNSTATS-Optionen angeben, wenn Sie Statistiken für eine Tabelle bzw. für Spalten erfassen. Je höher Sie diese Werte setzen, desto größer ist die Genauigkeit, mit der RUNSTATS Verteilungsstatistiken erstellt und aktualisiert. Allerdings erfordert eine größere Genauigkeit auch mehr Ressourcen, sowohl während der Ausführung von RUNSTATS als auch für die Speicherung in den Katalogtabellen.

Für die meisten Datenbanken empfiehlt sich ein Wert zwischen 10 und 100 für den Datenbankkonfigurationsparameter *num_freqvalues*. Im Idealfall sollten Häufigkeitsstatistikdaten so erstellt werden, dass die Häufigkeiten der verbleibenden Werte entweder einander annähernd gleich sind oder im Vergleich zu den Häufigkeiten der häufigsten Werte vernachlässigbar sind. Der Datenbankmanager erfasst unter

Umständen weniger als diese Anzahl, da diese Statistikdaten nur für Datenwerte erhoben werden, die mehr als einmal auftreten. Wenn Sie nur Quantilstatistiken erfassen müssen, setzen Sie den Parameter *num_freqvalues* auf den Wert 0.

Zur Einstellung der Anzahl von Quantilen geben Sie einen Wert zwischen 20 und 50 für den Datenbankkonfigurationsparameter *num_quantiles* an. Als grobe Faustregel zur Bestimmung der Anzahl von Quantilen gilt:

- Bestimmen Sie den maximalen Fehler, der bei der Schätzung der Anzahl von Zeilen jeder Bereichsabfrage tolerierbar ist, als Prozentwert P.
- Die Anzahl der Quantile sollte ca. 100/P sein, wenn es sich um ein BETWEEN-Vergleichselement handelt, und 50/P, wenn das Vergleichselement eine andere Art von Bereichsvergleichselement (<, <=, > oder >=) ist.

Zum Beispiel ergeben 25 Quantile einen maximalen Schätzfehler von 4% bei BETWEEN-Vergleichselementen und 2% bei Vergleichselementen mit ″>″. Allgemein sollten mindestens 10 Quantile angegeben werden. Mehr als 50 Quantile sind nur bei extrem ungleichmäßig verteilten Daten erforderlich. Wenn Sie nur Statistiken über die häufigsten Werte benötigen, setzen Sie den Parameter *num_quantiles* auf den Wert 0. Wenn Sie diesen Parameter auf den Wert „1″ setzen, werden keine Quantilstatistikdaten erfasst, da der gesamte Wertebereich in diesem Fall in nur einem Quantil liegt.

Zugehörige Konzepte:

- „Katalogstatistiken″ auf Seite 113
- „Katalogstatistiktabellen″ auf Seite 126
- „Verwendung der Verteilungsstatistiken durch das Optimierungsprogramm″ auf Seite 136
- „Erweiterte Beispiele zur Verwendung von Verteilungsstatistiken″ auf Seite 138

Zugehörige Tasks:

- „Erfassen von Katalogstatistiken″ auf Seite 117
- „Erfassen von Verteilungsstatistiken für bestimmte Spalten″ auf Seite 118

Zugehörige Referenzen:

- „num_freqvalues - Anzahl der häufigsten Werte″ auf Seite 506
- „num_quantiles - Anzahl der Quantile für Spalten″ auf Seite 507

Verwendung der Verteilungsstatistiken durch das Optimierungsprogramm

Das Optimierungsprogramm nutzt Verteilungsstatistiken zu besseren Abschätzungen des Aufwands für verschiedene mögliche Zugriffspläne zur Erfüllung von Abfragen.

Wenn Sie das Dienstprogramm RUNSTATS nicht mit der Klausel WITH DISTRIBUTION ausführen, enthalten die Katalogstatistiktabellen nur Informationen über die Größe der Tabelle, die höchsten und niedrigsten Werte in der Tabelle, den Grad der Clusterbildung der Tabelle in Bezug auf ihre Indizes und die Anzahl der unterschiedlichen Werte in den indexierten Spalten.

Sofern keine zusätzlichen Informationen über die Verteilung von Werten zwischen dem niedrigsten und dem höchsten Wert vorliegen, geht das Optimierungsprogramm davon aus, dass die Datenwerte gleichmäßig verteilt sind. Wenn die

Datenwerte erhebliche Abweichungen voneinander zeigen, in bestimmten Abschnitten des Wertebereichs Häufungen aufweisen oder einzelne Werte besonders zahlreich vorkommen, wählt das Optimierungsprogramm möglicherweise einen nicht optimalen Zugriffsplan aus.

Betrachten Sie das folgende Beispiel:

Das Optimierungsprogramm muss die Anzahl von Zeilen abschätzen, die einen Spaltenwert enthalten, der ein Gleichheits- oder Bereichsvergleichselement erfüllt, um den aufwandsgünstigsten Zugriffsplan auszuwählen. Je genauer die Schätzung ist, desto größer ist auch die Wahrscheinlichkeit, dass das Optimierungsprogramm den optimalen Zugriffsplan wählt. Betrachten Sie zum Beispiel die folgende Abfrage:

```
SELECT C1, C2
  FROM TABLE1
  WHERE C1 = 'NEW YORK'
  AND C2 <= 10
```

Nehmen Sie an, dass es einen Index sowohl für C1 als auch für C2 gibt. Ein möglicher Zugriffsplan besteht darin, über den Index für C1 alle Zeilen mit `C1 = 'NEW YORK'` abzurufen und anschließend jede abgerufene Zeile daraufhin zu überprüfen, ob `C2 <= 10` gilt. Ein alternativer Plan wäre, über den Index für C2 alle Zeilen mit `C2 <= 10` abzurufen und anschließend jede abgerufene Zeile daraufhin zu überprüfen, ob `C1 = 'NEW YORK'` gilt. Da der Hauptaufwand bei der Ausführung einer Abfrage in der Regel im Zusammenhang mit dem Abrufen der Zeilen anfällt, ist der Plan der beste, der die wenigsten Abrufe erfordert. Zur Auswahl dieses Plans ist eine Abschätzung der Anzahl von Zeilen erforderlich, die das jeweilige Vergleichselement erfüllen.

Wenn keine Verteilungsstatistiken verfügbar sind, jedoch RUNSTATS für die Tabelle ausgeführt wurde, sind die einzigen Informationen, die dem Optimierungsprogramm zur Verfügung stehen, der zweithöchste Wert (HIGH2KEY), der zweitniedrigste Wert (LOW2KEY), die Anzahl unterschiedlicher Werte (COLCARD) und die Anzahl von Zeilen für eine Spalte (CARD). Die Anzahl der Zeilen, die ein Gleichheitsvergleichselement oder ein Bereichsvergleichselement erfüllen, wird dann unter der Annahme abgeschätzt, dass die Häufigkeiten der Datenwerte in einer Spalte alle gleich und die Datenwerte gleichmäßig über das Intervall (LOW2KEY, HIGH2KEY) verteilt sind. Im Einzelnen wird die Anzahl der Zeilen, die ein Gleichheitsvergleichselement `C1 = KEY` erfüllen, mit dem Wert CARD /COLCARD abgeschätzt. Die Anzahl der Zeilen, die ein Bereichsvergleichselement `C1 BETWEEN KEY1 AND KEY2` erfüllen, wird nach folgender Formel abgeschätzt:

```
      KEY2 - KEY1
------------------  x CARD       (1)
 HIGH2KEY - LOW2KEY
```

Diese Schätzwerte sind nur dann realistisch, wenn die tatsächliche Verteilung der Datenwerte weitgehend gleichmäßig ist. Wenn keine Verteilungsstatistikdaten verfügbar sind und entweder die Häufigkeiten der Datenwerte grob von einander abweichen oder die Datenwerte in einigen wenigen Unterbereichen des Intervalls (LOW_KEY, HIGH_KEY) Häufungen bilden, können die Schätzwerte um Größenordnungen von der Realität abweichen, so dass das Optimierungsprogramm möglicherweise einen nicht optimalen Zugriffsplan wählt.

Wenn Verteilungsstatistikdaten verfügbar sind, können die oben beschriebenen Schätzfehler wesentlich verringert werden, indem die statistischen Daten zur Häufigkeit der Werte zur Berechnung der Anzahl von Zeilen, die ein Gleichheits-

vergleichselement erfüllen, und sowohl die statistischen Daten zur Häufigkeit der Werte als auch die Quantilstatistik zur Berechnung der Anzahl von Zeilen, die ein Bereichsvergleichselement erfüllen, herangezogen werden.

Zugehörige Konzepte:

- „Katalogstatistiken" auf Seite 113
- „Verteilungsstatistiken" auf Seite 133
- „Erweiterte Beispiele zur Verwendung von Verteilungsstatistiken" auf Seite 138

Zugehörige Tasks:

- „Erfassen von Verteilungsstatistiken für bestimmte Spalten" auf Seite 118

Erweiterte Beispiele zur Verwendung von Verteilungsstatistiken

Zur Erläuterung, wie das Optimierungsprogramm Verteilungsstatistiken nutzen kann, soll zunächst eine Abfrage betrachtet werden, die ein Gleichheitsvergleichselement der Form `C1 = KEY` enthält.

Beispiel für Statistiken über die Wertehäufigkeit

Wenn Statistiken zur Wertehäufigkeit verfügbar sind, kann das Optimierungsprogramm diese wie folgt zur Auswahl eines geeigneten Zugriffsplans verwenden:

- Wenn `KEY` einer der N häufigsten Werte ist, dann verwendet das Optimierungsprogramm die Häufigkeit von `KEY`, die im Katalog gespeichert ist.
- Wenn `KEY` nicht einer der N häufigsten Werte ist, schätzt das Optimierungsprogramm die Anzahl der Zeilen, die das Vergleichselement erfüllen, unter der Annahme ab, dass die nicht häufigen Werte (COLCARD - N) eine gleichmäßige Verteilung aufweisen. Das heißt, die Anzahl der Zeilen wird folgendermaßen abgeschätzt:

```
CARD - NUM_FREQ_ROWS
--------------------          (2)
   COLCARD - N
```

Dabei ist CARD die Anzahl von Zeilen in der Tabelle, COLCARD ist die Kardinalität der Spalte und COLCARD ist die Gesamtanzahl von Zeilen mit einem Wert, der gleich einem der N häufigsten Werte ist.

Betrachten Sie zum Beispiel eine Spalte (`C1`), für die sich die Häufigkeit der Datenwerte folgendermaßen darstellt:

Datenwert	Häufigkeit
1	2
2	3
3	40
4	4
5	1

Wenn Häufigkeitsstatistiken nur für die häufigsten Werte (d. h. N = 1) dieser Spalte verfügbar sind, ist die Anzahl der Zeilen in der Tabelle gleich 50 und die Spaltenkardinalität gleich 5. Das Vergleichselement `C1 = 3` wird exakt von 40 Zeilen erfüllt. Wenn das Optimierungsprogramm annimmt, dass die Datenwerte gleichmäßig verteilt sind, schätzt es die Anzahl der Zeilen, die das Vergleichselement erfüllen mit 50/5 = 10 ab, was einen Schätzfehler von -75% bedeutet.

Wenn das Optimierungsprogramm auf Häufigkeitsstatistiken zurückgreifen kann, wird die Anzahl der Zeilen mit 40 abgeschätzt und es entsteht in diesem Fall kein Fehler.

Betrachten Sie ein anderes Beispiel, in dem zwei Zeilen das Vergleichselement `C1 = 1` erfüllen. Ohne Häufigkeitsstatistiken wird die Anzahl der Zeilen, die das Vergleichselement erfüllen, auf 10 geschätzt. Mithin entsteht ein Fehler von 400 %. Die folgende Formel kann zur Berechnung des Schätzfehlers (in Prozent) herangezogen werden:

```
geschätzte Zeilen - tatsächliche Zeilen
---------------------------------------  X 100
          tatsächliche Zeilen
```

Bei Verwendung der Häufigkeitsstatistik (N = 1) schätzt das Optimierungsprogramm die Anzahl der Zeilen, die diesen Wert enthalten, anhand der oben gezeigten Formel (2). Zum Beispiel:

```
(50 - 40)
--------- = 3
 (5 - 1)
```

Der Fehler wird dabei um eine Größenordnung verringert, wie folgende Berechnung zeigt:

```
 3 - 2
------- = 50%
   2
```

Beispiel für Quantilstatistiken

In den folgenden Erläuterungen der Quantilstatistiken wird der Begriff „K-Quantile" verwendet. Das *K-Quantil* für eine Spalte ist der kleinste Datenwert `V`, so dass mindestens „K" Zeilen Datenwerte enthalten, die kleiner oder gleich `V` sind. Ein K-Quantil lässt sich berechnen, indem die Zeilen in der Spalte nach aufsteigenden Datenwerten sortiert werden. Das K-Quantil ist der Datenwert in der K-ten Zeile der sortierten Spalte.

Wenn Quantilstatistiken verfügbar sind, kann das Optimierungsprogramm die Anzahl von Zeilen besser abschätzen, die ein Bereichsvergleichselement erfüllen, wie in folgenden Beispielen veranschaulicht wird. Betrachten Sie eine Spalte (`C`), die folgende Werte enthält:

C
0,0
5,1
6,3
7,1
8,2
8,4
8,5
9,1
93,6
100,0

Nehmen Sie an, dass K-Quantile für K = 1, 4, 7 und 10 wie folgt zur Verfügung stehen:

K	K-Quantil
1	0,0
4	7,1
7	8,5
10	100,0

Betrachten Sie zunächst das Vergleichselement `C <= 8,5`. Für die oben angegebenen Daten erfüllen exakt sieben Zeilen dieses Vergleichselement. Unter der Annahme einer gleichmäßigen Datenverteilung und unter Verwendung der oben angegeben Formel (1), wobei KEY1 durch LOW2KEY ersetzt wird, wird die Anzahl der Zeilen, die das Vergleichselement erfüllen, folgendermaßen abgeschätzt:

```
 8,5 - 5,1
---------- x 10 *= 0
93,6 - 5,1
```

Die Notation `*=` bedeutet „annähernd gleich". Der Fehler bei dieser Schätzung ist annähernd -100%.

Wenn Quantilstatistiken verfügbar sind, schätzt das Optimierungsprogramm die Anzahl der Zeilen, die das gleiche Vergleichselement (`C <= 8,5`) erfüllen ab, indem es 8,5 als höchsten Wert eines der K-Quantile feststellt und die Anzahl der Zeilen ermittelt, indem es den entsprechenden Wert von K, also 7, verwendet. In diesem Fall reduziert sich der Fehler auf 0.

Betrachten Sie nun das Vergleichselement `C <= 10`. Dieses Vergleichselement wird von exakt acht Zeilen erfüllt. Wenn das Optimierungsprogramm von einer gleichmäßigen Datenwertverteilung ausgehen und Formel (1) verwenden muss, schätzt es die Anzahl von Zeilen, die das Vergleichselement erfüllen, auf 1, was einen Fehler von -87,5% ergibt.

Im Unterschied zum vorigen Beispiel ist der Wert 10 keiner der gespeicherten K-Quantile. Allerdings kann das Optimierungsprogramm mit Hilfe von Quantilen die Anzahl von Zeilen, die das Vergleichselement erfüllen, als `r_1 + r_2` abschätzen, wobei `r_1` die Anzahl von Zeilen ist, die das Vergleichselement `C <= 8,5` erfüllen, und `r_2` die Anzahl von Zeilen ist, die das Vergleichselement `C > 8,5 AND C <= 10` erfüllen. Wie im vorigen Beispiel gilt `r_1 = 7`. Zur Abschätzung von `r_2` verwendet das Optimierungsprogramm die lineare Interpolation:

```
        10 - 8,5
r_2 *= ---------- x (Anzahl Zeilen mit Wert > 8,5 und <= 100,0)
       100 - 8,5
       10 - 8,5
r_2 *= ---------- x (10 - 7)
       100 - 8,5
       1,5
r_2 *= ---- x (3)
       91,5
r_2 *= 0
```

Die abschließende Schätzung ist r_1 + r_2 *= 7, und der Fehler beträgt nur -12,5 %.

Quantile verbessern die Schätzgenauigkeit in den obigen Beispielen deswegen, weil die realen Datenwerte „Häufungen" im Bereich von 5 bis 10 bilden, aber die Standardformeln zur Schätzung von einer gleichmäßigen Verteilung der Werte zwischen 0 und 100 ausgehen.

Die Verwendung von Quantilen erhöht auch die Genauigkeit, wenn es wesentliche Unterschiede in den Häufigkeiten verschiedener Datenwerte gibt. Betrachten Sie eine Spalte, die Datenwerte mit den folgenden Häufigkeiten enthält:

Datenwert	Häufigkeit
20	5
30	5
40	15
50	50
60	15
70	5
80	5

Angenommen, es stehen K-Quantile zur Verfügung für K = 5, 25, 75, 95 und 100:

K	K-Quantil
5	20
25	40
75	50
95	70
100	80

Nehmen Sie außerdem an, dass statistische Häufigkeitsdaten für die 3 häufigsten Werte verfügbar sind.

Betrachten Sie das Vergleichselement `C BETWEEN 20 AND 30`. An der Verteilung der Datenwerte können Sie sehen, dass genau 10 Zeilen das Vergleichselement erfüllen. Unter der Annahme einer gleichmäßigen Datenverteilung und unter Verwendung der Formel (1) wird die Anzahl der Zeilen, die das Vergleichselement erfüllen, wie folgt abgeschätzt:

```
30 - 20
-------  x 100 = 25
70 - 30
```

Diese Abschätzung enthält einen Fehler von 150%.

Unter Verwendung der Häufigkeitsstatistik und der Quantile wird die Anzahl der Zeilen, die das Vergleichselement erfüllen, als `r_1 + r_2` abgeschätzt, wobei `r_1` die Anzahl der Zeilen ist, die das Vergleichselement `C = 20` erfüllen, und `r_2` die Anzahl der Zeilen ist, die das Vergleichselement `C > 20 AND C <= 30` erfüllen. Bei Verwendung der Formel (2) wird `r_1` folgendermaßen abgeschätzt:

```
100 - 80
-------- = 5
  7 - 3
```

`r_2` wird mit linearer Interpolation folgendermaßen abgeschätzt:

```
    30 - 20
    ------- x (Anzahl Zeilen mit Wert > 20 und <= 40)
    40 - 20
    30 - 20
= ------- x (25 - 5)
    40 - 20
= 10,
```

Als endgültiger Schätzwert ergibt sich 15, wodurch der Schätzfehler um den Faktor 3 verringert wird.

Zugehörige Konzepte:

- „Katalogstatistiken" auf Seite 113
- „Verteilungsstatistiken" auf Seite 133
- „Verwendung der Verteilungsstatistiken durch das Optimierungsprogramm" auf Seite 136
- „Regeln zur manuellen Aktualisierung von Verteilungsstatistiken" auf Seite 153

Zugehörige Tasks:

- „Erfassen von Verteilungsstatistiken für bestimmte Spalten" auf Seite 118

Detaillierte Indexstatistiken

Wenn Sie RUNSTATS für Indizes mit der Klausel DETAILED ausführen, erfassen Sie statistische Informationen zu Indizes, mit deren Hilfe das Optimierungsprogramm abschätzen kann, wie viele Datenseitenabrufe für die unterschiedlichen Pufferpoolgrößen erforderlich würden. Diese Zusatzinformationen unterstützen das Optimierungsprogramm bei einer besseren Abschätzung des Aufwands für den Zugriff auf eine Tabelle über einen Index.

Anmerkung: Wenn Sie detaillierte Indexstatistiken erfassen, benötigt RUNSTATS mehr Zeit und erfordert mehr Speicher und CPU-Kapazitäten. Für die Option SAMPLED DETAILED, bei der eine Berechnung von Daten nur für eine statistisch relevante Anzahl von Einträgen erfolgt, sind 2 MB Statistikzwischenspeicher erforderlich. Ordnen Sie wegen dieses zusätzlichen Speicherbedarfs dem Datenbankkonfigurationsparameter *stat_heap_sz* weitere 488 4-KB-Seiten zu. Wenn der Zwischenspeicher zu klein erscheint, gibt RUNSTATS einen Fehler zurück, bevor das Dienstprogramm versucht, Statistikdaten zu erfassen.

Die DETAILED-Statistiken PAGE_FETCH_PAIRS und CLUSTERFACTOR werden nur erfasst, wenn die Tabelle eine ausreichende Größe hat (ca. 25 Seiten). In diesem Fall hat CLUSTERFACTOR einen Wert zwischen 0 und 1, während CLUSTERRATIO den Wert -1 (nicht erfasst) hat. Für Tabellen, die kleiner als 25 Seiten sind, hat CLUSTERFACTOR den Wert -1 (nicht erfasst) und CLUSTERRATIO einen Wert zwischen 0 und 100, auch wenn die Klausel DETAILED für einen Index für diese Tabelle angegeben wird.

Mit Hilfe der Klausel DETAILED werden komprimierte Informationen über die Anzahl der physischen E/A-Operationen bereitgestellt, die für den Zugriff auf die Datenseiten einer Tabelle erforderlich werden, wenn eine vollständige Indexsuche bei verschiedenen Puffergrößen ausgeführt wird. Das Dienstprogramm RUNSTATS sucht die Seiten des Index ab und modelliert dabei die verschiedenen Puffergrößen und sammelt Schätzwerte darüber, wie häufig eine Fehlseitenbedingung auftritt. Wenn zum Beispiel nur eine Pufferseite verfügbar ist, führt jede neue Seite, auf die der Index verweist, zu einer Fehlseite. In einem ungünstigeren Fall könnte jede Zeile auf eine andere Seite verweisen, was zur gleichen Anzahl von E/A-Operationen wie Zeilen in der indexierten Tabelle führt. Im entgegengesetzten Extremfall, wenn der Puffer groß genug ist, um die gesamte Tabelle (abhängig von der maximalen Puffergröße) aufzunehmen, werden alle Tabellenseiten nur einmal gelesen. Die Anzahl der physischen E/A-Operationen ist infolgedessen eine monotone, nicht steigende Funktion der Puffergröße.

Die statistischen Informationen ermöglichen außerdem feinere Schätzwerte für den Grad der Clusterbildung der Tabellenzeilen in Bezug auf die Indexreihenfolge. Je

geringer die Clusterbildung der Tabellenzeilen in Relation zum Index ist, desto mehr E/A-Operationen sind für den Zugriff auf Tabellenzeilen über den Index erforderlich. Das Optimierungsprogramm zieht sowohl die Puffergröße als auch den Grad der Clusterbildung bei der Abschätzung des Aufwands für den Zugriff auf eine Tabelle über einen Index in Betracht.

Die detaillierten Indexstatistiken sollten erfasst werden, wenn Abfragen auf Spalten zugreifen, die nicht im Index enthalten sind. Darüber hinaus sollten detaillierte Indexdaten in folgenden Fällen verwendet werden:
- Die Tabelle hat mehrere Indizes ohne Clustering mit verschiedenen Graden von Clusterbildung.
- Der Grad der Clusterbildung unter den Schlüsselwerten ist nicht gleichmäßig.
- Die Werte im Index werden ungleichmäßig aktualisiert.

Eine Beurteilung dieser Bedingungen ist ohne Vorwissen bzw. ohne eine zwangsweise durchgeführte Indexsuche unter verschiedenen Puffergrößen und eine Überwachung der sich ergebenden physischen E/A-Operationen schwierig. Die Methode des geringsten Aufwands zur Feststellung, ob eine dieser Situationen auftritt, besteht wahrscheinlich darin, die DETAILED-Statistikdaten für einen Index zu erfassen, zu untersuchen und zu behalten, wenn die resultierenden Wertepaare für PAGE_FETCH_PAIRS nicht linear sind.

Zugehörige Konzepte:
- „Katalogstatistiken” auf Seite 113
- „Katalogstatistiktabellen” auf Seite 126

Zugehörige Tasks:
- „Erfassen von Katalogstatistiken” auf Seite 117
- „Erfassen von Indexstatistiken” auf Seite 119

Unterelementstatistiken

Wenn Tabellen Spalten enthalten, die durch Leerzeichen getrennte Unterfelder oder Unterelemente enthalten, und Abfragen auf diese Spalten in WHERE-Klauseln verweisen, sollten Sie Unterelementstatistiken erfassen, um die Auswahl der besten Zugriffspläne zu gewährleisten.

Nehmen Sie zum Beispiel an, eine Datenbank enthält eine Tabelle namens DOCUMENTS, deren Zeilen jeweils ein Dokument beschreiben, und nehmen Sie ferner an, DOCUMENTS enthält eine Spalte namens KEYWORDS, die eine Liste relevanter Schlüsselwörter für das Dokument enthält, die zum Abrufen von Texten verwendet werden. Die Spalte KEYWORDS könnte z. B. folgende Werte enthalten:

```
'Datenbank Simulation Analytisch Business Intelligence'
'Simulation Modell Fruchtfliege Reproduktion Temperatur'
'Forstwirtschaft Fichte Boden Erosion Regen'
'Wald Temperatur Boden Niederschlag Brand'
```

In diesem Beispiel besteht jede Spalte aus 5 Unterelementen, von denen jedes ein Wort (das Schlüsselwort) ist und durch ein Leerzeichen von den anderen Wörtern getrennt ist.

Bei Abfragen, die für solche Spalten das Vergleichselement LIKE mit dem Universalplatzhalterzeichen % verwenden:

```
SELECT .... FROM DOCUMENTS WHERE KEYWORDS LIKE '%Simulation%'
```

ist es für das Optimierungsprogramm häufig von Nutzen, wenn es einige grundlegende Statistikdaten zur Unterelementstruktur der Spalte zur Verfügung hat.

Die folgenden Statistiken werden erfasst, wenn Sie das Dienstprogramm RUNSTATS mit der Klausel LIKE STATISTICS ausführen:

SUB_COUNT
Die durchschnittliche Anzahl der Unterelemente.

SUB_DELIM_LENGTH
Die durchschnittliche Länge der einzelnen Begrenzer, die Unterelemente voneinander trennen. Unter einem Begrenzer ist in diesem Zusammenhang ein Leerzeichen oder mehrere aufeinander folgende Leerzeichen zu verstehen.

Im Beispiel der Spalte KEYWORDS hat SUB_COUNT den Wert 5, und SUB_DELIM_LENGTH den Wert 1, da jeder Begrenzer ein einzelnes Leerzeichen ist.

Die Registriervariable DB2®_LIKE_VARCHAR beeinflusst die Art und Weise, wie das Optimierungsprogramm ein Vergleichselement der folgenden Form behandelt:

```
COLUMN LIKE '%xxxxxx'
```

(Dabei steht *xxxxxx* für eine beliebige Zeichenfolge) Das heißt, ein beliebiges LIKE-Vergleichselement, dessen Suchwert mit einem Prozentzeichen (%) beginnt. (Der Suchwert kann, muss aber nicht mit % enden.) Solche Vergleichselemente werden in der Folge als LIKE-Vergleichselemente mit Platzhalter bezeichnet. Das Optimierungsprogramm muss für alle Vergleichselemente abschätzen, wie viele Zeilen dem Vergleichselement entsprechen. Bei LIKE-Vergleichselementen mit Platzhaltern nimmt das Optimierungsprogramm an, dass die verglichene Spalte (COLUMN) eine Reihe miteinander verketteter Elemente enthält, und schätzt die Länge jedes Elements ausgehend von der Länge der Zeichenfolge ohne führende und abschließende %-Zeichen ab.

Sie können die Werte der Unterelementstatistiken durch eine Abfrage der Tabelle SYSIBM.SYSCOLUMNS untersuchen. Zum Beispiel:

```
select substr(NAME,1,16), SUB_COUNT, SUB_DELIM_LENGTH
   from sysibm.syscolumns where tbname = 'DOCUMENTS'
```

Anmerkung: Bei Verwendung der Klausel LIKE STATISTICS benötigt das Dienstprogramm eventuell mehr Zeit. Bei einer Tabelle mit fünf Zeichenspalten zum Beispiel kann die Ausführung von RUNSTATS zwischen 15 und 40 % länger dauern, wenn die Optionen DETAILED und DISTRIBUTION nicht verwendet werden. Wird die Option DETAILED oder DISTRIBUTION angegeben, ist der Mehraufwand prozentual u. U. geringer, auch wenn der Mehraufwand absolut betrachtet derselbe ist. Wenn Sie die Verwendung dieser Option beabsichtigen, sollten Sie zwischen diesem Mehraufwand und den Verbesserungen der Abfrageleistung abwägen.

Zugehörige Konzepte:

- „Katalogstatistiken" auf Seite 113
- „Katalogstatistiktabellen" auf Seite 126

Zugehörige Tasks:

- „Erfassen von Katalogstatistiken" auf Seite 117
- „Erfassen von Verteilungsstatistiken für bestimmte Spalten" auf Seite 118

Vom Benutzer aktualisierbare Katalogstatistiken

Dieser Abschnitt beschreibt die Katalogstatistikdaten, die Benutzer manuell aktualisieren können, und enthält Richtlinien für solche manuellen Änderungen. In einigen Fällen werden die betreffenden Statistikdaten nicht durch das Dienstprogramm RUNSTATS gesammelt, sondern müssen manuell hinzugefügt werden. In anderen Fällen können Sie gesammelte Statistikdaten in eine Testdatenbank importieren und die gesammelten Daten für einen speziellen Zweck ändern, um zum Beispiel eine Produktionsdatenbank zu Experimentalzwecken zu modellieren.

Warnung: In einer Produktionsdatenbank sollten Sie die von RUNSTATS gesammelten Daten auf keinen Fall manuell aktualisieren. Anderenfalls könnten sich schwerwiegende Leistungsprobleme ergeben.

Statistiken für benutzerdefinierte Funktionen

Zur Erstellung statistischer Informationen für benutzerdefinierte Funktionen (UDFs) editieren Sie die Katalogsicht SYSSTAT.FUNCTIONS. Wenn UDF-Statistiken verfügbar sind, können sie vom Optimierungsprogramm zur Abschätzung des Aufwands für verschiedene Zugriffspläne verwendet werden. Das Dienstprogramm RUNSTATS erfasst keine Statistiken für benutzerdefinierte Funktionen. Sind keine Statistiken verfügbar, enthalten die Statistikspalten den Wert -1, und das Optimierungsprogramm verwendet Standardwerte, die von einer einfachen benutzerdefinierten Funktion ausgehen.

Die folgende Tabelle enthält Informationen zu den Statistikspalten, für die Sie Schätzwerte angeben können, um die Leistung zu verbessern:

Tabelle 25. Funktionsstatistiken (SYSCAT.FUNCTIONS und SYSSTAT.FUNCTIONS)

Statistik	Beschreibung
IOS_PER_INVOC	Geschätzte Anzahl der Schreib-/Leseanforderungen, die jedes Mal ausgeführt werden, wenn eine Funktion ausgeführt wird
INSTS_PER_INVOC	Geschätzte Anzahl der Maschinenanweisungen, die jedes Mal ausgeführt werden, wenn eine Funktion ausgeführt wird
IOS_PER_ARGBYTE	Geschätzte Anzahl der Schreib-/Leseanforderungen, die für jedes Eingabeargumentbyte ausgeführt werden
INSTS_PER_ARGBYTES	Geschätzte Anzahl der Maschinenanweisungen, die für jedes Eingabeargumentbyte ausgeführt werden
PERCENT_ARGBYTES	Geschätzter Durchschnittsprozentwert der Eingabeargumentbyte, die von der Funktion tatsächlich verarbeitet werden
INITIAL_IOS	Geschätzte Anzahl der Schreib-/Leseanforderungen, die nur beim ersten/letzten Aufruf der Funktion ausgeführt werden
INITIAL_INSTS	Geschätzte Anzahl der Maschinenanweisungen, die nur beim ersten/letzten Aufruf der Funktion ausgeführt werden

Tabelle 25. Funktionsstatistiken (SYSCAT.FUNCTIONS und SYSSTAT.FUNCTIONS) (Forts.)

Statistik	Beschreibung
CARDINALITY	Geschätzte Anzahl von Zeilen, die von einer Tabellenfunktion generiert werden

Betrachten Sie zum Beispiel eine benutzerdefinierte Funktion (EU_SHOE), die eine amerikanische Schuhgröße in die entsprechende europäische Schuhgröße umwandelt. (Diese beiden Schuhgrößen könnten benutzerdefinierte Datentypen (UDTs) haben.) Für diese UDF könnten Sie die Statistikspalten mit folgenden Werten versehen:

- Für INSTS_PER_INVOC könnte als Wert die geschätzte Anzahl der Maschineninstruktionen festgelegt werden, die zu folgenden Operationen erforderlich sind:
  - Aufrufen der Funktion EU_SHOE
  - Initialisieren der Ausgabezeichenfolge
  - Rückgabe des Ergebnisses
- Für INSTS_PER_ARGBYTE könnte als Wert die geschätzte Anzahl der Maschineninstruktionen festgelegt werden, die zur Umwandlung der Eingabezeichenfolge in die europäische Schuhgröße erforderlich ist.
- Der Wert für PERCENT_ARGBYTES könnte auf 100 gesetzt werden, was anzeigt, dass die gesamte Eingabezeichenfolge umzuwandeln ist.
- Die Werte für INITIAL_INSTS, IOS_PER_INVOC, IOS_PER_ARGBYTE und INITIAL_IOS könnten jeweils auf 0 gesetzt werden, da diese benutzerdefinierte Funktion lediglich Berechnungen ausführt.

PERCENT_ARGBYTES würde für eine Funktion verwendet, die nicht immer die gesamte Eingabezeichenfolge verarbeitet. Ein Beispiel wäre eine benutzerdefinierte Funktion (LOCATE), an die zwei Argumente als Eingabe übergeben werden und die die Anfangsposition des ersten Vorkommens des ersten Arguments innerhalb des zweiten Arguments als Ergebnis zurückliefert. Nehmen Sie an, dass die Länge des ersten Arguments ausreichend klein ist, um im Vergleich zum zweiten Argument kaum eine Rolle zu spielen, und im Durchschnitt 75% des zweiten Arguments zum Auffinden des ersten durchsucht werden. Aufgrund dieser Informationen sollte der Wert für PERCENT_ARGBYTES auf 75 gesetzt werden. Die obige Abschätzung eines Durchschnitts von 75% fußt dabei auf folgenden zusätzlichen Annahmen:

- In der Hälfte der Fälle wird das erste Argument gar nicht gefunden, d. h., das gesamte zweite Argument wird durchsucht.
- Die Wahrscheinlichkeit, dass das erste Argument innerhalb des zweiten Arguments auftritt, ist an allen Stellen gleich groß, d. h., in den Fällen, in denen das erste Argument überhaupt gefunden wird, muss im Durchschnitt die Hälfte des zweiten Arguments durchsucht werden.

Sie können die Spalten INITIAL_INSTS bzw. INITIAL_IOS zur Eintragung der geschätzten Anzahl von Maschineninstruktionen bzw. Schreib-/Leseanforderungen verwenden, die nur beim ersten bzw. letzten Aufruf der Funktion ausgeführt werden, um zum Beispiel den Aufwand zur Einrichtung eines Arbeitspufferbereichs zu vermerken.

Informationen über Ein-/Ausgaben und Instruktionen, die von einer benutzerdefinierten Funktion verursacht werden, erhalten Sie über die Ausgaben des Compilers für die Programmiersprache bzw. der vom Betriebssystem bereitgestellten Überwachungsprogramme.

Zugehörige Konzepte:

- „Katalogstatistiktabellen" auf Seite 126
- „Allgemeine Regeln zur manuellen Aktualisierung von Katalogstatistiken" auf Seite 151

Katalogstatistiken zu Modellierung und Fallstudien

Sie können die statistischen Informationen in den Systemkatalogen abweichend vom tatsächlichen Status von Tabellen und Indizes ändern, um verschiedene mögliche Änderungen an der Datenbank zu Planungszwecken zu untersuchen. Aufgrund dieser Aktualisierbarkeit ausgewählter Systemkatalogstatistiken haben Sie folgende Möglichkeiten:

- Sie können die Abfrageleistung auf einem Entwicklungssystem unter Verwendung wirklichkeitsnaher Systemstatistiken eines geschäftlich genutzten Systems nachmodellieren.
- Sie können Fallstudien („Was wäre, wenn?") für die Abfrageleistung durchführen und die Ergebnisse analysieren.

Nehmen Sie keine manuellen Aktualisierungen an den Statistiken eines Produktionssystems vor. Anderenfalls wählt das Optimierungsprogramm möglicherweise nicht den besten Zugriffsplan für Abfragen in der Produktionsumgebung aus, die dynamisches SQL enthalten.

Voraussetzungen

Sie müssen über eine explizite DBADM-Berechtigung für die Datenbank verfügen, um Statistiken für Tabellen und Indizes und ihre Komponenten modifizieren zu können. Das heißt, Ihre Benutzer-ID muss in der Tabelle SYSCAT.DBAUTH mit der Berechtigung DBADM eingetragen sein. Durch die Zugehörigkeit zu einer DBADM-Gruppe wird diese Berechtigung nicht explizit erteilt. Ein DBADM-Benutzer kann die Statistikzeilen für alle Benutzer abrufen und kann SQL-Anweisungen UPDATE an den für das Schema SYSSTAT definierten Sichten ausführen, um die Werte dieser Statistikspalten zu aktualisieren.

Ein Benutzer ohne DBADM-Berechtigung kann nur die Zeilen sehen, die statistische Daten für Objekte enthalten, für die er das Zugriffsrecht CONTROL hat. Wenn Sie keine DBADM-Berechtigung besitzen, können Sie die Statistiken für einzelne Datenbankobjekte ändern, wenn Sie die folgenden Zugriffsrechte für jedes Objekt haben:

- Explizites Zugriffsrecht CONTROL für Tabellen. Sie können auch die Statistiken für Spalten und Indizes dieser Tabellen aktualisieren.
- Explizites Zugriffsrecht CONTROL für Kurznamen in einem System zusammengeschlossener Datenbanken. Sie können auch Statistiken für Spalten und Indizes für diese Kurznamen aktualisieren. Beachten Sie, dass die Aktualisierung nur lokale Metadaten betrifft (Tabellenstatistiken der Datenquelle werden nicht geändert). Diese Aktualisierungen betreffen nur die globale Zugriffsstrategie, die vom DB2®-Optimierungsprogramm generiert wird.
- Eigentumsrecht für benutzerdefinierte Funktionen (UDFs).

Das folgende Beispiel zeigt, wie die Tabellenstatistik für die Tabelle EMPLOYEE aktualisiert werden kann:

```
UPDATE SYSSTAT.TABLES
SET  CARD   = 10000,
     NPAGES = 1000,
```

```
      FPAGES = 1000,
      OVERFLOW = 2
WHERE TABSCHEMA = 'benutzer-id'
  AND TABNAME   = 'EMPLOYEE'
```

Bei der manuellen Aktualisierung von Katalogstatistiken ist Vorsicht geboten. Willkürliche Änderungen können die Leistung nachfolgender Abfragen ernsthaft beeinflussen. Auch in einer nicht in der Produktion eingesetzten Datenbank, die Sie zum Testen oder Modellieren verwenden, können Sie jede der folgenden Methoden anwenden, um Aktualisierungen, die Sie auf diese Tabellen angewendet haben, zu erneuern und die Statistiken in einen konsistenten Status zu bringen:

- Rückgängigmachen (ROLLBACK) der Arbeitseinheit, in der Änderungen durchgeführt wurden (unter der Annahme, dass die Arbeitseinheit noch nicht festgeschrieben wurde).
- Ausführen des Dienstprogramms RUNSTATS, um die Katalogstatistiken neu berechnen und aktualisieren zu lassen.
- Aktualisieren der Katalogstatistiken, um anzugeben, dass keine Statistikdaten gesammelt wurden. (Zum Beispiel wird durch den Wert -1 in der Spalte NPAGES angezeigt, dass keine Statistik zur Anzahl von Seiten erfasst wurde.)
- Ersetzen der Katalogstatistiken durch die Daten, die sie enthielten, bevor Änderungen durchgeführt wurden. Diese Methode ist nur möglich, wenn Sie das Tool *db2look* zum Erfassen der Statistiken vor Ihren Änderungen ausgeführt haben.

In einigen Fällen kann es vorkommen, dass das Optimierungsprogramm einen bestimmten statistischen Wert oder eine Kombination von Werten als ungültig erkennt. In diesem Fall verwendet es die Standardwerte und gibt eine Warnung aus. Situationen dieser Art sind jedoch selten, da der Hauptteil der Gültigkeitsprüfungen bei der Aktualisierung der Statistiken durchgeführt wird.

Zugehörige Konzepte:

- „Katalogstatistiken" auf Seite 113
- „Katalogstatistiktabellen" auf Seite 126
- „Statistiken für benutzerdefinierte Funktionen" auf Seite 145
- „Statistiken zur Modellierung von Produktionsdatenbanken" auf Seite 148

Statistiken zur Modellierung von Produktionsdatenbanken

Manchmal ist es wünschenswert, auf einem Testsystem einen Teil der Daten eines tatsächlich geschäftlich genutzten Systems nachzubilden. Jedoch sind die Zugriffspläne, die auf einem solchen Testsystem ausgewählt werden, nicht unbedingt dieselben wie die, die auf dem tatsächlich genutzten System gewählt würden, sofern nicht die Katalogstatistiken und die Konfigurationsparameter auf dem Testsystem so aktualisiert werden, dass sie mit denen auf dem Produktionssystem übereinstimmen.

Das Tool *db2look* kann für die Produktionsdatenbank ausgeführt werden, um die Aktualisierungsanweisungen (UPDATE-Anweisungen) zu generieren, die erforderlich sind, um die Katalogstatistiken der Testdatenbank in Übereinstimmung mit denen der Produktionsdatenbank zu bringen. Diese Aktualisierungsanweisungen können mit Hilfe des Programms *db2look* mit der Option `-m` (mimic-Modus) generiert werden. In diesem Fall generiert das Tool *db2look* eine Prozedur für den Befehlsprozessor, die alle Anweisungen enthält, die erforderlich sind, um die Katalogstatistiken der Produktionsdatenbank nachzubilden. Dies kann bei der Analyse von SQL-Anweisungen mit Hilfe von Visual Explain in einer Testumgebung nützlich sein.

Sie können Datenbankdatenobjekte einschließlich Tabellen, Sichten, Indizes und andere Objekte in einer Datenbank wieder erstellen, indem Sie die DDL-Anweisungen mit dem Befehl *db2look -e* extrahieren. Sie können die Prozedur für den Befehlsprozessor, die mit diesem Befehl erstellt wurde, in einer anderen Datenbank ausführen, um die Datenbank neu zu erstellen. Sie können die Option `-e` und die Option `-m` zusammen in einer Prozedur verwenden, die die Datenbank neu erstellt und die Statistikdaten festlegt.

Nach der Ausführung der von `db2look` erstellten Aktualisierungsanweisungen im Testsystem kann das Testsystem zur Prüfung der Zugriffspläne verwendet werden, die in der Produktionsdatenbank generiert werden sollen. Da das Optimierungsprogramm die Art und Konfiguration der Tabellenbereiche zur Abschätzung der E/A-Aufwände verwendet, muss das Testsystem über dieselbe Tabellenbereichsanordnung bzw. -konfiguration verfügen. Das heißt, sie muss dieselbe Anzahl von Behältern desselben Typs, SMS oder DMS, haben.

Das Tool *db2look* befindet sich im Unterverzeichnis *bin*.

Weitere Informationen zur Verwendung dieses Tools erhalten Sie, wenn Sie folgenden Befehl in eine Befehlszeile eingeben:

```
db2look -h
```

Darüber hinaus bietet die Steuerzentrale eine Schnittstelle zum Programm *db2look* unter dem Namen „DDL generieren für Objektname". Die Steuerzentrale ermöglicht eine Integration der Ergebnisdatei aus dem Dienstprogramm in die Prozedurzentrale. Sie können außerdem den Befehl *db2look* über die Steuerzentrale zeitlich terminieren. Ein Unterschied bei der Verwendung der Steuerzentrale besteht darin, dass eine Analyse nur einer Tabelle durchgeführt werden kann, während in einem Aufruf des Befehls *db2look* bis zu 30 Tabellen analysiert werden können. Beachten Sie darüber hinaus, dass LaTex- und Graphical-Ausgaben über die Steuerzentrale nicht unterstützt werden.

Sie können das Dienstprogramm `db2look` auch für eine OS/390- oder z/OS-Datenbank ausführen. Das Dienstprogramm `db2look` extrahiert die DDL- und UPDATE-Anweisungen für die Statistiken von OS/390-Objekten. Diese Funktion ist sehr nützlich, wenn Sie OS/390- oder z/OS-Objekte extrahieren und in einer DB2® UDB-Datenbank (UDB - Universal Database) erneut erstellen möchten.

Zwischen den Statistikdaten von DB2 UDB und denen von OS/390 gibt es einige Unterschiede. Das Dienstprogramm `db2look` führt die entsprechenden Umsetzungen von DB2 für OS/390 oder z/OS auf DB2 UDB aus, wenn dies zutreffend ist, und setzt die Statistikdaten von DB2 UDB, für die in DB2 für OS/390 keine Entsprechung vorhanden ist, auf den Standardwert (-1). Im Folgenden wird beschrieben, wie das Dienstprogramm `db2look` die Statistikdaten von DB2 für OS/390 oder z/OS den Statistikdaten von DB2 UDB zuordnet. In den nachfolgenden Erläuterungen steht jeweils „UDB_x" für eine Statistikdatenspalte von DB2 UDB. „S390_x" steht für eine Statistikdatenspalte von DB2 für OS/390 oder z/OS.

1. Statistikdaten auf Tabellenebene.

   UDB_CARD = S390_CARDF
   UDB_NPAGES = S390_NPAGES

   Es gibt keinen Parameter S390_FPAGES. DB2 für OS/390 oder z/OS verfügt jedoch über einen anderen Parameter für Statistikdaten mit dem Namen PCTPAGES, der den Prozentsatz von aktiven Tabellenbereichsseiten darstellt, die

Zeilen der Tabelle enthalten. Daher kann der Wert des Parameters UDB_FPAGES auf der Basis von S390_NPAGES und S390_PCTPAGES wie folgt berechnet werden:

```
UDB_FPAGES=(S390_NPAGES * 100)/S390_PCTPAGES
```

Es gibt keinen Parameter S390_OVERFLOW, der UDB_OVERFLOW zugeordnet werden kann. Daher wird dieser Wert vom Dienstprogramm `db2look` einfach auf den Standardwert gesetzt:

```
UDB_OVERFLOW=-1
```

2. Statistikdaten auf Spaltenebene.

   UDB_COLCARD = S390_COLCARDF
   UDB_HIGH2KEY = S390_HIGH2KEY
   UDB_LOW2KEY = S390_LOW2KEY

   Es gibt keinen Parameter S390_AVGCOLLEN, der UDB_AVGCOLLEN zugeordnet werden kann. Daher wird dieser Wert vom Dienstprogramm `db2look` einfach auf den Standardwert gesetzt:

   ```
   UDB_AVGCOLLEN=-1
   ```

3. Statistikdaten auf Indexebene.

   UDB_NLEAF = S390_NLEAF
   UDB_NLEVELS = S390_NLEVELS
   UDB_FIRSTKEYCARD= S390_FIRSTKEYCARD
   UDB_FULLKEYCARD = S390_FULLKEYCARD
   UDB_CLUSTERRATIO= S390_CLUSTERRATIO

   Die anderen Statistikdaten, für die es keine OS/390- oder z/OS-Entsprechungen gibt, werden einfach auf den Standardwert gesetzt. Dies heißt Folgendes:

   ```
   UDB_FIRST2KEYCARD = -1
   UDB_FIRST3KEYCARD = -1
   UDB_FIRST4KEYCARD  = -1
   UDB_CLUSTERFACTOR = -1
   UDB_SEQUENTIAL_PAGES = -1
   UDB_DENSITY  = -1
   ```

4. Spaltenverteilungsstatistiken.

   In der Tabelle SYSIBM.SYSCOLUMNS unter DB2 für OS/390 oder z/OS gibt es zwei Typen von Statistikdaten. Der Typ „F" für häufige (Frequent) Werte und der Typ „C" für Kardinalität (Cardinality). Nur Einträge des Typs „F" gelten für DB2 für UDB. Dies sind auch die Einträge, die berücksichtigt werden.

   ```
   UDB_COLVALUE = S390_COLVALUE
   UDB_VALCOUNT = S390_FrequencyF * S390_CARD
   ```

   Außerdem gibt es in der Tabelle SYSIBM.SYSCOLUMNS unter DB2 für OS/390 keine Spalte SEQNO. Da eine solche Spalte für DB2 für UDB erforderlich ist, generiert das Dienstprogramm `db2look` eine solche Spalte automatisch.

Zugehörige Konzepte:

- „Katalogstatistiken" auf Seite 113
- „Katalogstatistiktabellen" auf Seite 126
- „Katalogstatistiken zu Modellierung und Fallstudien" auf Seite 147
- „Allgemeine Regeln zur manuellen Aktualisierung von Katalogstatistiken" auf Seite 151

Zugehörige Referenzen:

- „db2look - DB2 Statistics and DDL Extraction Tool Command" in *Command Reference*

Allgemeine Regeln zur manuellen Aktualisierung von Katalogstatistiken

Die wichtigste Regel, die bei einer Aktualisierung der Katalogstatistiken zu beachten ist, ist die Sicherstellung, dass gültige Werte, Wertebereiche und Formate der verschiedenen Statistiken in den Statistiksichten gespeichert werden. Darüber hinaus muss die Konsistenz der Beziehungen zwischen verschiedenen Statistiken gewahrt bleiben.

Zum Beispiel muss der Wert für COLCARD in der Sicht SYSSTAT.COLUMNS kleiner sein als der für CARD in der Sicht SYSSTAT.TABLES (die Anzahl der unterschiedlichen Werte in einer Spalte kann nicht größer sein als die Anzahl der Zeilen). Nehmen Sie an, Sie wollen den Wert für COLCARD von 100 auf 25 und den Wert für CARD von 200 auf 50 verringern. Wenn Sie die Sicht SYSCAT.TABLES zuerst aktualisieren, sollten Sie eine Fehlernachricht empfangen (da CARD kleiner als COLCARD würde). Die richtige Reihenfolge wäre, zuerst den Wert für COLCARD in der Sicht SYSCAT.COLUMNS und anschließend den Wert für CARD in der Sicht SYSSTAT.TABLES zu aktualisieren. Der Fall stellt sich umgekehrt dar, wenn Sie den Wert von COLCARD von 100 auf 250 und den Wert für CARD von 200 auf 300 erhöhen wollen. In diesem Fall müssten Sie zuerst den Wert CARD und anschließend den Wert COLCARD aktualisieren.

Wenn ein Konflikt zwischen einer aktualisierten Statistik und einer anderen Statistik festgestellt wird, wird eine Fehlernachricht ausgegeben. Jedoch werden vielleicht nicht immer Fehler gemeldet, wenn Konflikte auftreten. In einigen Fällen können die Konflikte nur schwer festgestellt und als Fehler gemeldet werden, besonders wenn die beiden zusammengehörigen Statistiken in verschiedenen Katalogen gespeichert sind. Aus diesem Grund sollten Sie solche Konflikte umsichtig vermeiden.

Die folgenden allgemeinen Prüfungen sollten vor der Aktualisierung einer Katalogstatistik durchgeführt werden:

1. Numerische Statistikdaten müssen -1 bzw. größer oder gleich null (0) sein.
2. Numerische Statistikdaten, die Prozentwerte darstellen (z. B. CLUSTERRATIO in der Katalogsicht SYSSTAT.INDEXES), müssen zwischen 0 und 100 liegen.

Anmerkung: Bei Zeilentypen sind die statistischen Daten auf Tabellenebene für NPAGES, FPAGES und OVERFLOW für eine untergeordnete Tabelle nicht aktualisierbar.

Zugehörige Konzepte:

- „Katalogstatistiktabellen" auf Seite 126
- „Statistiken für benutzerdefinierte Funktionen" auf Seite 145
- „Katalogstatistiken zu Modellierung und Fallstudien" auf Seite 147
- „Statistiken zur Modellierung von Produktionsdatenbanken" auf Seite 148
- „Regeln zur manuellen Aktualisierung von Spaltenstatistiken" auf Seite 152
- „Regeln zur manuellen Aktualisierung von Verteilungsstatistiken" auf Seite 153
- „Regeln zur manuellen Aktualisierung von Tabellen- und Kurznamenstatistiken" auf Seite 154
- „Regeln zur manuellen Aktualisierung von Indexstatistiken" auf Seite 154

Regeln zur manuellen Aktualisierung von Spaltenstatistiken

Bei der Aktualisierung von Statistikdaten in der Katalogsicht SYSSTAT.COLUMNS sind folgende Regeln zu beachten.

- Folgen Sie bei der manuellen Aktualisierung der Werte für die Spalten HIGH2KEY und LOW2KEY in der Sicht SYSSTAT.COLUMNS der Funktionsweise der generierten Werte:
  - Die Werte für HIGH2KEY und LOW2KEY müssen gültige Werte für den Datentyp der entsprechenden Benutzerspalte sein.
  - Die Länge der Werte für HIGH2KEY, LOW2KEY muss entweder 33 oder die maximale Länge des Datentyps der Zielspalte betragen, je nachdem, welcher der beiden Werte kleiner ist. Dies schließt zusätzliche Anführungszeichen nicht mit ein, durch die die Länge der Zeichenfolge auf 68 anwachsen kann. Dies bedeutet, dass nur die ersten 33 Zeichen des Werts in der entsprechenden Benutzerspalte bei der Bestimmung der Werte für HIGH2KEY, LOW2KEY in Betracht gezogen werden.
  - Die HIGH2KEY/LOW2KEY-Werte werden so gespeichert, dass sie in der SET-Klausel einer UPDATE-Anweisung und ohne Manipulation an Aufwandsberechnungen verwendet werden können. Für Zeichenfolgen bedeutet dies, dass einfache Anführungszeichen am Anfang und am Ende der Zeichenfolge angefügt und ein zusätzliches Anführungszeichen für jedes bereits in der Zeichenfolge vorhandene Anführungszeichen hinzugefügt wird. Beispiele von Benutzerspaltenwerten und ihren entsprechenden Werten für HIGH2KEY,LOW2KEY finden Sie in der folgenden Tabelle.

Tabelle 26. HIGH2KEY- und LOW2KEY-Werte für Datentypen

Datentyp in Benutzerspalte	Benutzerdaten	Entsprechender HIGH2KEY-, LOW2KEY-Wert
INTEGER	–12	–12
CHAR	abc	'abc'
CHAR	ab'c	'ab''c'

  - Der Wert für HIGH2KEY sollte größer als der Wert für LOW2KEY sein, wenn es mehr als drei unterschiedliche Werte in der entsprechenden Spalte gibt.
- Die Kardinalität einer Spalte (Statistik COLCARD in der Katalogsicht SYSSTAT.COLUMNS) kann nicht größer sein als die Kardinalität der zugehörigen Tabelle (Statistik CARD in der Katalogsicht SYSSTAT.TABLES).
- Die Anzahl der Nullwerte in einer Spalte (Statistik NUMNULLS in der Katalogsicht SYSSTAT.COLUMNS) kann nicht größer sein als die Kardinalität der zugehörigen Tabelle (Statistik CARD in der Katalogsicht SYSSTAT.TABLES).
- Für Spalten mit folgenden Datentypen werden keine Statistikdaten unterstützt: LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB.

Zugehörige Konzepte:

- „Katalogstatistiken" auf Seite 113
- „Katalogstatistiktabellen" auf Seite 126
- „Katalogstatistiken zu Modellierung und Fallstudien" auf Seite 147
- „Allgemeine Regeln zur manuellen Aktualisierung von Katalogstatistiken" auf Seite 151

Regeln zur manuellen Aktualisierung von Verteilungsstatistiken

Manuelle Aktualisierungen von Verteilungsstatistiken nehmen Sie nur zu dem Zweck vor, eine Produktionsdatenbank nachzumodellieren oder Fallstudien für eine künstlich konstruierte Datenbank durchzuführen. Nehmen Sie keine manuellen Aktualisierungen an den Verteilungsstatistiken einer Produktionsdatenbank vor. Stellen Sie sicher, dass alle Statistiken im Katalog konsistent sind. Im Einzelnen müssen die Katalogeinträge für Häufigkeitsstatistiken und Quantildaten für jede Spalte die folgenden Bedingungen erfüllen:

- Häufigkeitsstatistiken (in der Katalogsicht SYSSTAT.COLDIST). Dazu gehören:
  - Die Werte in der Spalte VALCOUNT müssen bei steigenden Werten für SEQNO unverändert bleiben oder abnehmen.
  - Die Anzahl der Werte in der Spalte COLVALUE muss kleiner oder gleich der Anzahl der unterschiedlichen Werte in der Spalte sein. Diese Anzahl wird in der Spalte COLCARD der Katalogsicht SYSSTAT.COLUMNS gespeichert.
  - Die Summe der Werte in der Spalte VALCOUNT muss kleiner oder gleich der Anzahl der Zeilen in der Spalte sein. Diese Anzahl wird in der Spalte CARD der Katalogsicht SYSSTAT.TABLES gespeichert.
  - In der Regel sollten die Werte in der Spalte COLVALUE zwischen dem zweithöchsten und dem zweitniedrigsten Datenwert für die Spalte liegen. Diese beiden Werte werden in der Spalte HIGH2KEY bzw. LOW2KEY der Katalogsicht SYSSTAT.COLUMNS gespeichert. Es kann nur einen häufigen Wert geben, der größer als der Wert für HIGH2KEY ist, und einen, der kleiner als der Wert für LOW2KEY ist.
- Quantile (in der Katalogsicht SYSSTAT.COLDIST). Dazu gehören:
  - Die Werte in der Spalte COLVALUE müssen bei steigenden Werten für SEQNO unverändert bleiben oder abnehmen.
  - Die Werte in der Spalte VALCOUNT müssen bei steigenden Werten für SEQNO steigen.
  - Der größte Wert in der Spalte COLVALUE muss einen entsprechenden Eintrag in der Spalte VALCOUNT haben, der gleich der Anzahl von Zeilen in der Spalte ist.
  - In der Regel sollten die Werte in der Spalte COLVALUE zwischen dem zweithöchsten und dem zweitniedrigsten Datenwert für die Spalte liegen. Diese beiden Werte werden in der Spalte HIGH2KEY bzw. LOW2KEY der Katalogsicht SYSSTAT.COLUMNS gespeichert.

Nehmen Sie an, es stehen Verteilungsstatistikdaten für eine Spalte C1 mit „Z" Zeilen zur Verfügung, und Sie möchten die Statistiken so modifizieren, dass sie einer Spalte mit identischen relativen Proportionen der Datenwerte, aber mit „(F x Z)" Zeilen entsprechen. Um die Größenordnung der Häufigkeitsstatistiken um einen Faktor `F` zu erhöhen, muss jeder Eintrag der Spalte VALCOUNT mit `F` multipliziert werden. Um die Quantilwerte ebenfalls um einen Faktor `F` zu vergrößern, muss jeder Eintrag der Spalte VALCOUNT mit `F` multipliziert werden. Wenn Sie diese Regeln nicht beachten, kann das Optimierungsprogramm den falschen Filterfaktor verwenden, was bei der Ausführung der Abfrage zu nicht vorhersehbaren Auswirkungen auf die Leistung führen kann.

Zugehörige Konzepte:

- „Katalogstatistiken" auf Seite 113
- „Katalogstatistiktabellen" auf Seite 126
- „Katalogstatistiken zu Modellierung und Fallstudien" auf Seite 147
- „Allgemeine Regeln zur manuellen Aktualisierung von Katalogstatistiken" auf Seite 151

Regeln zur manuellen Aktualisierung von Tabellen- und Kurznamenstatistiken

Die einzigen statistischen Werte, die Sie in der Katalogsicht SYSTAT.TABLES aktualisieren können, sind CARD, FPAGES, NPAGES und OVERFLOW sowie für MDC-Tabellen ACTIVE_BLOCKS. Beachten Sie dabei Folgendes:

1. Der Wert für CARD der Tabelle muss größer als oder gleich allen auf diese Tabelle bezogenen Werten für COLCARD in der Katalogsicht SYSSTAT.COLUMNS sein.
2. Der Wert für CARD muss größer als der Wert für NPAGES sein.
3. Der Wert für FPAGES muss größer als der Wert für NPAGES sein.
4. Der Wert für NPAGES muss kleiner oder gleich jedem Wert für den Abrufteil der Wertepaare in der Spalte PAGE_FETCH_PAIRS für jeden Index sein (unter der Annahme, dass diese Statistik für den Index relevant ist).
5. Der Wert für CARD darf nicht kleiner oder gleich irgendeinem Wert für den Seitenabruf in den Wertepaaren der Spalte PAGE_FETCH_PAIRS für jeden Index sein (unter der Annahme, dass diese Statistik für den Index relevant ist).

Wenn Sie in einem System zusammengeschlossener Datenbanken arbeiten, gehen Sie bei der Erstellung bzw. Aktualisierung von Statistiken für einen Kurznamen über eine ferne Sicht sehr vorsichtig vor. Die statistischen Informationen, wie zum Beispiel die Anzahl von Zeilen, die von diesem Kurznamen zurückgegeben werden, entsprechen möglicherweise nicht dem realen Aufwand zur Auswertung dieser Sicht und können das DB2®-Optimierungsprogramm irreführen. Fälle, die von Aktualisierungen der Statistiken profitieren können, sind ferne Sichten, die für eine einzelne ferne Tabelle ohne Anwendung von Spaltenfunktionen in der SELECT-Liste definiert wurden. Komplexe Sichten erfordern möglicherweise einen komplexen Optimierungsprozess, der die Optimierung jeder einzelnen Abfrage erforderlich macht. Ziehen Sie stattdessen die Erstellung lokaler Sichten über Kurznamen in Betracht, damit das DB2-Optimierungsprogramm in der Lage ist, den Aufwand für die Sicht exakter abzuschätzen.

Zugehörige Konzepte:

- „Katalogstatistiken" auf Seite 113
- „Katalogstatistiktabellen" auf Seite 126
- „Katalogstatistiken zu Modellierung und Fallstudien" auf Seite 147
- „Allgemeine Regeln zur manuellen Aktualisierung von Katalogstatistiken" auf Seite 151

Regeln zur manuellen Aktualisierung von Indexstatistiken

Bei der Aktualisierung der Statistiken in der Katalogsicht SYSSTAT.INDEXES sind folgende Regeln zu beachten:

1. Die Werte für PAGE_FETCH_PAIRS (in der Katalogsicht SYSSTAT.INDEXES) müssen folgende Regeln erfüllen:
   - Einzelne Werte in der Statistik PAGE_FETCH_PAIRS müssen durch eine Folge von Leerzeichenbegrenzern getrennt werden.

- Einzelne Werte in der Statistik PAGE_FETCH_PAIRS dürfen eine Länge von 10 Stellen nicht überschreiten und müssen kleiner als der größte ganzzahlige Wert (MAXINT = 2147483647) sein.
- Es muss immer einen gültigen Wert für PAGE_FETCH_PAIRS geben, wenn für CLUSTERFACTOR ein Wert größer null (0) angegeben ist.
- Es müssen sich genau 11 Wertepaare in einer einzelnen Statistik PAGE-_FETCH_PAIR befinden.
- Die Einträge für die Puffergrößen in PAGE_FETCH_PAIRS müssen eine aufsteigende Wertefolge bilden.
- Kein Wert für die Puffergröße in einem Eintrag für PAGE_FETCH_PAIRS darf den Wert MIN(NPAGES, 524287) bei 32-Bit-Betriebssystemen bzw. MIN(NPAGES, 2147483647) bei 64-Bit-Betriebssystemen überschreiten, wobei NPAGES die Anzahl der Seiten in der entsprechenden Tabelle (in der Katalogsicht SYSSTAT.TABLES) ist.
- Einträge für „Seitenabrufe" in PAGE_FETCH_PAIRS müssen in absteigender Wertefolge angegeben werden, wobei kein einzelner Eintrag für „Seitenabrufe" kleiner als NPAGES sein darf. Einträge für „Seitenabrufe" in PAGE-_FETCH_PAIRS können nicht größer als der Wert der Statistik CARD (Kardinalität) der zugehörigen Tabelle sein.
- Wenn der Wert für die Puffergröße in zwei aufeinander folgenden Paaren übereinstimmt, muss der Wert für den Seitenabruf (Page fetch) in beiden Paaren ebenfalls übereinstimmen (in der Katalogsicht SYSSTAT.TABLES).

Eine gültige Aktualisierung der Wertefolge für PAGE_FETCH_PAIRS ist zum Beispiel:

```
PAGE_FETCH_PAIRS =
 '100 380 120 360 140 340 160 330 180 320 200 310 220 305 240 300
  260 300 280 300 300 300'
```

mit

```
NPAGES = 300
CARD   = 10000
CLUSTERRATIO  =  -1
CLUSTERFACTOR = 0.9
```

2. Die Werte für CLUSTERRATIO und CLUSTERFACTOR (in der Katalogsicht SYSSTAT.INDEXES) müssen folgende Regeln erfüllen:
   - Gültige Werte für CLUSTERRATIO (Clusterverhältnis) sind -1 bzw. Werte zwischen 0 und 100.
   - Gültige Werte für CLUSTERFACTOR (Clusterfaktor) sind -1 bzw. Werte zwischen 0 und 1.
   - Es muss immer mindestens einer der Werte für CLUSTERRATIO und CLUSTERFACTOR gleich -1 sein.
   - Wenn für CLUSTERFACTOR ein positiver Wert angegeben ist, müssen gültige Statistikdaten in der Spalte PAGE_FETCH_PAIR enthalten sein.
3. Die folgenden Regeln gelten für die Werte der Statistiken FIRSTKEYCARD, FIRST2KEYCARD, FIRST3KEYCARD, FIRST4KEYCARD und FULLKEYCARD:
   - Der Wert für FIRSTKEYCARD muss gleich dem Wert für FULLKEYCARD für einen einspaltigen Index sein.
   - Der Wert für FIRSTKEYCARD muss gleich dem Wert COLCARD (in SYSSTAT.COLUMNS) für die entsprechende Spalte sein.
   - Wenn einige dieser Indexstatistikwerte nicht relevant sind, sollten Sie sie auf den Wert -1 setzen. Wenn Sie beispielsweise einen Index mit nur drei Spalten haben, setzen Sie FIRST4KEYCARD auf den Wert -1.
   - Für mehrspaltige Indizes gilt die folgende Beziehung, wenn alle Statistikwerte relevant sind:

     ```
     FIRSTKEYCARD <= FIRST2KEYCARD <= FIRST3KEYCARD <= FIRST4KEYCARD
        <= FULLKEYCARD <= CARD
     ```

4. Die folgenden Regeln gelten für SEQUENTIAL_PAGES und DENSITY:
   - Gültige Werte für SEQUENTIAL_PAGES sind -1 bzw. Werte zwischen 0 und NLEAF.
   - Gültige Werte für DENSITY sind -1 bzw. Werte zwischen 0 und 100.

Zugehörige Konzepte:

- „Katalogstatistiken" auf Seite 113
- „Katalogstatistiktabellen" auf Seite 126
- „Katalogstatistiken zu Modellierung und Fallstudien" auf Seite 147
- „Allgemeine Regeln zur manuellen Aktualisierung von Katalogstatistiken" auf Seite 151

Kapitel 6. Der SQL-Compiler

Wenn eine SQL-Abfrage kompiliert wird, wird eine Reihe von Schritten ausgeführt, bevor der ausgewählte Zugriffsplan entweder ausgeführt oder im Systemkatalog gespeichert wird.

In einer Umgebung mit einer partitionierten Datenbank finden alle Operationen, die vom SQL-Compiler an einer SQL-Abfrage ausgeführt werden, in der Datenbankpartition statt, zu der Sie die Verbindung herstellen. Vor der Ausführung wird die kompilierte Abfrage an alle Datenbankpartitionen in der Datenbank gesendet.

Die Themen in diesem Kapitel enthalten weitere Informationen zur Kompilierung und Optimierung von SQL-Anweisungen durch den SQL-Compiler.

Der SQL-Compilerprozess

Der SQL-Compiler führt mehrere Schritte aus, um einen Zugriffsplan zu erstellen, der ausgeführt werden kann. Diese Schritte werden in der folgenden Abbildung dargestellt und in den anschließenden Abschnitten beschrieben. Beachten Sie, dass einige Schritte nur für Abfragen in einer zusammengeschlossenen Datenbank durchgeführt werden.

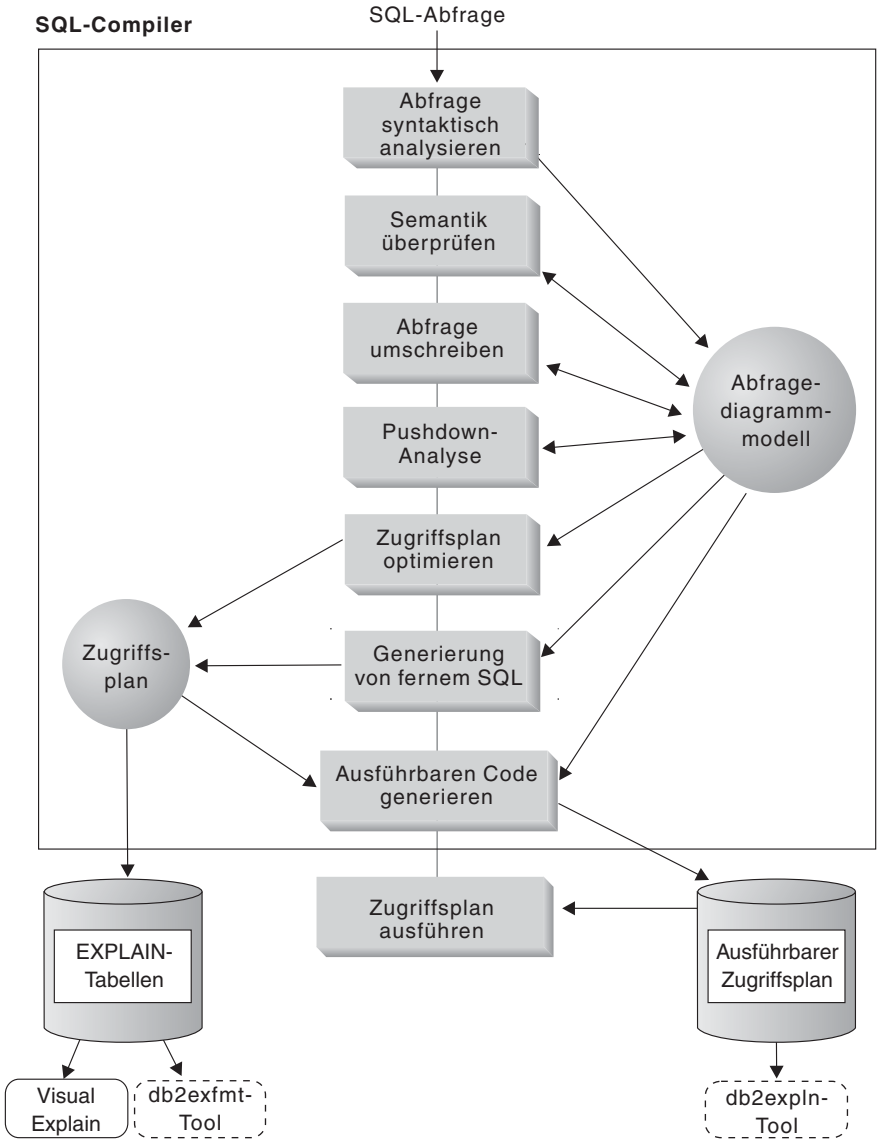


Abbildung 12. Vom SQL-Compiler ausgeführte Schritte

Abfragediagrammmodell

Das *Abfragediagrammmodell* ist eine interne, im Speicher befindliche Datenbank, welche die Abfrage im Verlauf der in den folgenden Schritten beschriebenen Verarbeitung darstellt:

1. **Abfrage analysieren**

   Der SQL-Compiler analysiert die SQL-Abfrage, um die Gültigkeit der Syntax zu überprüfen. Wenn Syntaxfehler festgestellt werden, beendet der SQL-Compiler die Verarbeitung und gibt die entsprechende SQL-Fehlernachricht an die Anwendung zurück, die die SQL-Anweisung übergeben hat. Wenn die Analyse abgeschlossen ist, wird eine interne Darstellung der Abfrage erstellt und im Abfragediagrammmodell gespeichert.

2. **Semantik prüfen**

   Der Compiler stellt sicher, dass keine Inkonsistenzen zwischen Teilen der Anweisung bestehen. Ein einfaches Beispiel für eine Semantikprüfung ist die

Überprüfung durch den Compiler, ob eine für die Skalarfunktion YEAR angegebene Spalte mit dem Datentyp DATETIME (Datum/Uhrzeit) definiert wurde.

Der Compiler fügt außerdem die Bedingungssemantik in das Abfragediagrammmodell ein, zu der die Auswirkungen der referenziellen Integritätsbedingungen, der Prüfungen auf Integritätsbedingungen in Tabellen, der Auslöser und der Sichten gehören. Das Abfragediagrammmodell enthält die gesamte Semantik der Abfrage, einschließlich der Abfrageblöcke, Unterabfragen, Korrelationen, abgeleiteten Tabellen, Ausdrücken, Datentypen, Datentypumsetzungen, Codepageumwandlungen und der Partitionierungsschlüssel.

3. **Abfrage umschreiben**

   Der Compiler verwendet die im Abfragediagrammmodell gespeicherte globale Semantik, um die Abfrage in eine Form umzusetzen, die leichter optimiert werden kann, und speichert das Ergebnis im Abfragediagrammmodell.

   Zum Beispiel kann der Compiler ein Vergleichselement verschieben und somit die Ebene ändern, auf der es angewandt wird, um dadurch potenziell die Abfrageleistung zu erhöhen. Diese Art der Verschiebung einer Operation wird als allgemeine Vergleichselementverschiebung (*General Predicate Pushdown*) bezeichnet. In einer Umgebung mit partitionierten Datenbanken sind die folgenden Abfrageoperationen etwas rechenintensiver:

   - Spaltenberechnungen (Aggregation)
   - Umverteilen von Zeilen
   - Korrelierte Unterabfragen, d h. Unterabfragen, die einen Verweis auf eine Spalte einer Tabelle enthalten, die sich außerhalb der Unterabfrage befindet

   Für einige Abfragen in einer partitionierten Umgebung kann eine Dekorrelation im Rahmen des Umschreibens der Abfrage erfolgen.

4. **Pushdown-Analyse (zusammengeschlossene Datenbanken)**

   Die Hauptfunktion dieses Schrittes ist, dem Optimierungsprogramm eine Empfehlung zu liefern, ob eine Operation an eine Datenquelle verschoben und fern ausgewertet werden kann, was als *Pushdown* bezeichnet wird. Diese Art von Pushdown-Aktivität ist für Datenquellenabfragen spezifisch und bildet eine Erweiterung zu den allgemeinen Operationen der Vergleichselementverschiebung.

   Dieser Schritt wird nur ausgeführt, wenn Abfragen für zusammengeschlossene Datenbanken ausgeführt werden.

5. **Zugriffsplan optimieren**

   Mit dem Abfragediagrammmodell als Eingabe generiert die Optimierungskomponente des Compilers zahlreiche alternative Ausführungspläne zur Erfüllung der Abfrage. Das Optimierungsprogramm schätzt den Ausführungsaufwand mit Hilfe der Statistiken für Tabellen, Indizes, Spalten und Funktionen für jeden der verschiedenen Pläne ab. Dann wählt es den Plan mit dem geringsten geschätzten Ausführungsaufwand aus. Das Optimierungsprogramm verwendet das Abfragediagrammmodel, um die Abfragesemantik zu analysieren und Informationen zu einer Vielzahl von Faktoren, einschließlich Indizes, Basistabellen, abgeleitete Tabellen, Unterabfragen, Korrelationen und Rekursion, zu erhalten.

   Das Optimierungsprogramm kann außerdem eine andere Art von Verschiebeoperation (Pushdown) in Betracht ziehen, nämlich für *Spaltenberechnungen und Sortierungen*. Die Leistung kann erhöht werden, wenn die Auswertung dieser Operationen an die Komponente der Datenverwaltungsservices (Data Management Services) verschoben werden kann.

   Das Optimierungsprogramm berücksichtigt auch, ob es Pufferpools verschiedener Größen gibt, wenn es die Auswahl der Seitengröße festlegt. Der Faktor,

dass die Umgebung eine partitionierte Datenbank enthält, wird ebenso berücksichtigt wie die Möglichkeit, den ausgewählten Plan für eine potenzielle abfrageinterne Parallelität in einer symmetrischen Multiprozessorumgebung (SMP-Umgebung) auszulegen. Diese Informationen werden vom Optimierungsprogramm zur Auswahl des am besten geeigneten Zugriffsplans für die Abfrage verwendet.

Die Ausgabe dieses Schritts des Compilers ist ein Zugriffsplan. Dieser Zugriffsplan stellt die Informationen bereit, die in den EXPLAIN-Tabellen erfasst werden. Die Informationen, die zur Generierung des Zugriffsplans dienten, können mit einer Momentaufnahme der EXPLAIN-Einrichtung erfasst werden.

6. **Generierung von fernem SQL (zusammengeschlossene Datenbanken)**

   Der endgültige Plan, der vom Optimierungsprogramm gewählt wird, kann aus einer Reihe von Schritten bestehen, die an einer fernen Datenquelle ausgeführt werden. Für Operationen, die an der jeweiligen Datenquelle ausgeführt werden, erstellt der Schritt der Generierung von fernem SQL eine effiziente SQL-Anweisung, die auf der SQL-Version der Datenquelle beruht.

7. **„Ausführbaren" Code generieren**

   Im letzten Schritt verwendet der Compiler den Zugriffsplan und das Abfragediagrammmodell, um einen ausführbaren Zugriffsplan oder Abschnitt für die Abfrage zu erstellen. Bei dieser Generierung des Codes werden Informationen des Abfragediagrammmodells verwendet, um eine Wiederholung der Ausführung von Ausdrücken zu vermeiden, die für eine Abfrage nur einmal berechnet werden müssen. Diese Art der Optimierung ist beispielsweise für Umwandlungen von Codepages und die Verwendung von Hostvariablen möglich.

   Damit eine Abfrageoptimierung bzw. -reoptimierung für statische und dynamische SQL-Anweisungen, die Hostvariablen, Sonderregister oder Parametermarken enthalten, ausgeführt werden kann, binden Sie das Paket mit der Bindeoption REOPT. Wenn diese Bindeoption verwendet wird, wird der Zugriffspfad für eine SQL-Anweisung, die zu diesem Paket gehört und Hostvariablen, Parametermarken oder Sonderregister enthält, mit den Werten dieser Variablen, und nicht mit Standardschätzwerten, die der Compiler auswählt, optimiert. Diese Optimierung erfolgt bei der Ausführung der Abfrage, wenn die Werte verfügbar sind.

   Informationen über die Zugriffspläne für statisches SQL werden in den Systemkatalogtabellen gespeichert. Wenn das Paket ausgeführt wird, verwendet der Datenbankmanager die in den Systemkatalogtabellen gespeicherten Informationen, um festzulegen, wie auf die Daten zugegriffen werden soll, und Ergebnisse für eine Abfrage bereitzustellen. Diese Informationen werden vom Tool *db2expln* verwendet.

Anmerkung: Führen Sie RUNSTATS in angemessenen Intervallen für Tabellen aus, die häufig geändert werden. Das Optimierungsprogramm benötigt aktuelle Statistikinformationen zu den Tabellen und ihren Daten, um die effizientesten Zugriffspläne zu generieren. Binden Sie Ihre Anwendung erneut, um die aktualisierten Statistiken zu nutzen. Wenn das Dienstprogramm RUNSTATS nicht ausgeführt wird oder das Optimierungsprogramm annimmt, dass RUNSTATS für leere oder fast leere Tabellen ausgeführt wurde, kann das Optimierungsprogramm entweder Standardwerte verwenden oder versuchen, bestimmte Statistikdaten mit Hilfe der Anzahl der Dateiseiten (FPAGES) abzuleiten, die zum Speichern der Tabelle auf dem Plattenspeicher verwendet werden. Die Gesamtanzahl belegter Blöcke wird in der Spalte ACTIVE_BLOCKS gespeichert.

Zugehörige Konzepte:

- „Methoden zum Umschreiben von Abfragen und Beispiele" auf Seite 164
- „Datenzugriffsmethoden" auf Seite 174
- „Terminologie für Vergleichselemente" auf Seite 181
- „Verknüpfungen" auf Seite 184
- „Auswirkungen des Sortierens und Gruppierens" auf Seite 201
- „Optimierungsstrategien für partitionsinterne Parallelität" auf Seite 203
- „Gespeicherte Abfragetabellen" auf Seite 207
- „Richtlinien zur Analyse, wo eine Abfrage für zusammengeschlossene Datenbanken ausgewertet wird" auf Seite 215
- „Advantages of Deferred Binding" in *Application Development Guide: Programming Client Applications*
- „Optimierungsstrategien für MDC-Tabellen" auf Seite 206

Konfigurationsparameter mit Einfluss auf die Abfrageoptimierung

Verschiedene Konfigurationsparameter wirken sich auf die Auswahl des Zugriffsplans durch den SQL-Compiler aus. Viele von ihnen gelten für eine Datenbank mit Einzelpartition, während einige nur für eine partitionierte Datenbank gelten. In einer partitionierten Datenbank sollten die Werte, die für die einzelnen Parameter verwendet werden, für alle Partitionen jeweils 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 den Befehl FLUSH PACKAGE CACHE aus.

Wenn in einem System zusammengeschlossener Datenbanken 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 zusammengeschlossenen Datenbank keine Seiten aus Datenquellen im Cache zwischen. Datenquellen sind die Datenbankverwaltungssysteme (DBMSs) und Daten in einem System zusammengeschlossener Datenbanken. 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 für DB2® Universal Database verfügbaren Ressourcen die Leistung verbessern.

Die folgenden Konfigurationsparameter bzw. Faktoren wirken sich auf die Auswahl des Zugriffsplans durch den SQL-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 BUFFERPOOLS und, in partitionierten Datenbanken, in den Systemkatalogtabellen BUFFERPOOLDBPARTITION.

Der Ein-/Ausgabeaufwand für das Lesen der Tabellen kann sich auf folgende Bereiche auswirken:
– Wie zwei Tabellen verknüpft werden.
– Ob ein Index ohne Clusterbildung zum Lesen der Daten verwendet wird.

- Grad der Parallelität (dft_degree)

  Der Konfigurationsparameter `dft_degree` gibt die Parallelität durch Bereitstellen eines Standardwerts für das Sonderregister CURRENT DEGREE und Bindeoption DEGREE an. Der Wert 1 bedeutet keine partitionsinterne Parallelität. Der Wert -1 bedeutet, dass das Optimierungsprogramm den Grad der partitionsinternen Parallelität anhand der Anzahl von Prozessoren und der Art der Abfrage bestimmt.

- Standardabfrageoptimierungsklasse (dft_queryopt)

  Obwohl Sie beim Kompilieren von SQL-Abfragen eine Abfrageoptimierungsklasse angeben können, ist es vielleicht sinnvoll, einen Standardoptimierungsgrad zu definieren.

  Anmerkung: Eine partitionsinterne Parallelverarbeitung findet nur statt, wenn Sie sie durch das Definieren des Datenbankkonfigurationsparameters *intra_parallel* aktivieren.

- Durchschnittliche Anzahl aktiver Anwendungen (avg_appls)

  Mit Hilfe des Parameters *avg_appls* versucht das SQL-Optimierungsprogramm zu ermitteln, wie viel vom Pufferpool zur Laufzeit für den ausgewählten Zugriffsplan verfügbar ist. Höhere Werte für diesen Parameter können das Optimierungsprogramm dahin gehend beeinflussen, dass es Zugriffspläne auswählt, der 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 Zwischenspeicher für Sortierlisten 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 Tabelle 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 wird als „über eine Pipe geleitet (piped)" betrachtet, wenn sie keine temporäre Tabelle zur Speicherung der endgültigen, sortierten Liste von Daten erforderlich macht. Das heißt, dass die Ergebnisse der Sortierung in einem einzigen sequenziellen Zugriff gelesen werden können. Über eine Pipe geleitete Sortierungen führen zu einer besseren Leistung als nicht über eine Pipe geleitete und werden daher, wenn möglich, verwendet.

  Bei der Auswahl eines Zugriffsplans schätzt das Optimierungsprogramm den Aufwand der Sortieroperationen, einschließlich der Möglichkeiten, eine Sortierung über eine Pipe zu leiten, folgendermaßen ab:
  – Abschätzen der Menge der zu sortierenden Daten

  - Bestimmen mit Hilfe des Parameters *sortheap*, ob genügend Speicherbereich für die Sortierung mit Pipe zur Verfügung steht.
- Maximaler Speicher für Sperrenliste (locklist) und maximale Anzahl Sperren pro Anwendung (maxlocks)

  Wenn die Isolationsstufe **RR** (Wiederholtes Lesen) verwendet wird, berücksichtigt das Optimierungsprogramm die Werte der Parameter *locklist* und *maxlocks*, um zu bestimmen, 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 den Systemaufwand, der mit einer Sperreneskalation während der Ausführung der Abfrage verbunden wäre.
- CPU-Geschwindigkeit (cpuspeed)

  Der Parameter für die CPU-Geschwindigkeit wird vom SQL-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 Migration der Datenbank automatisch auf einen 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. Mit Hilfe dieses Parameters können Sie eine andere Hardwareumgebung modellieren, um die Zugriffspläne zu ermitteln, die für die andere Umgebung ausgewählt würden. Wenn DB2 den Wert dieses automatischen Konfigurationsparameters wieder neu berechnen soll, setzen Sie den Parameter auf den Wert -1.
- Anweisungszwischenspeicher (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-Anweisungen ausgeführt wird.

  Wenn der Wert für den Parameter *stmtheap* nicht groß genug ist, empfangen Sie eventuell eine SQL-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.
- Maximaler Grad der Parallelität bei Abfragen (max_querydegree)

  Wenn der Parameter *max_querydegree* den Wert „ANY" hat, wählt das Optimierungsprogramm den Grad der Parallelität für die Anwendung aus. Ist ein anderer Wert als „ANY" definiert, legt der vom Benutzer angegebene Wert den Grad der Parallelität für die Anwendung fest.
- Kommunikationsbandbreite (comm_bandwidth)

  Die Kommunikationsbandbreite wird vom Optimierungsprogramm verwendet, um den Zugriffsplan zu bestimmen. Das Optimierungsprogramm verwendet den Wert dieses Parameters, um den Aufwand für bestimmte Operationen zwischen den Datenbankpartitionsservern einer partitionierten Datenbank abzuschätzen.

Zugehörige Konzepte:

- „Der SQL-Compilerprozess" auf Seite 157
- „Datenzugriffsmethoden" auf Seite 174

- „Optimierungsstrategien für partitionsinterne Parallelität" auf Seite 203
- „Optimierungsstrategien für MDC-Tabellen" auf Seite 206

Zugehörige Referenzen:
- „max_querydegree - Maximaler Grad der Parallelität bei Abfragen" auf Seite 522
- „comm_bandwidth - Kommunikationsbandbreite" auf Seite 529
- „sortheap - Zwischenspeichergröße für Sortierlisten" auf Seite 418
- „locklist - Maximaler Speicher für Sperrenliste" auf Seite 399
- „maxlocks - Maximale Anzahl Sperren pro Anwendung" auf Seite 433
- „stmtheap - Größe des Anweisungszwischenspeichers" auf Seite 420
- „cpuspeed - CPU-Geschwindigkeit" auf Seite 530
- „avg_appls - Durchschnittliche Anzahl aktiver Anwendungen" auf Seite 443
- „dft_degree - Standardgrad der Parallelität" auf Seite 502

Umschreiben von Abfragen

Dieser Abschnitt erläutert die Methoden, mit denen das Optimierungsprogramm Abfragen zur Verbesserung der Leistung umschreiben kann.

Methoden zum Umschreiben von Abfragen und Beispiele

Während der Phase des Umschreibens der Abfrage setzt der SQL-Compiler SQL-Anweisungen in eine Form um, die leichter optimiert werden und infolgedessen die möglichen Zugriffspfade verbessern kann. Das Umschreiben von Abfragen ist besonders wichtig für komplexe Abfragen, zu denen auch Abfragen mit zahlreichen Unterabfragen oder Verknüpfungen zählen. Tools zur Generierung von Abfragen erstellen häufig diese sehr komplexen Arten von Abfragen.

Zur Beeinflussung der Anzahl von Regeln für das Umschreiben von Abfragen, die auf eine SQL-Anweisung angewandt werden, ändern Sie die Optimierungsklasse. Einige der Ergebnisse dieses Umschreibens der Abfrage können Sie mit Hilfe der EXPLAIN-Einrichtung oder über Visual Explain anzeigen.

Abfragen können durch die folgenden drei Hauptmethoden umgeschrieben werden:

- **Zusammenfügen von Operationen**

  Zur Generierung der Abfrage in einer Form, die möglichst wenige Operationen, insbesondere SELECT-Operationen, besitzt, schreibt der SQL-Compiler Abfragen um, um Abfrageoperationen zusammenzufügen. Die folgenden Beispiele zeigen einige der Operationen, die zusammengefügt werden können:
  - Beispiel - Zusammenfügen von Sichten

    Eine SELECT-Anweisung, die Sichten verwendet, kann die Verknüpfungsfolge der Tabelle einschränken und zudem überflüssige Verknüpfungen von Tabellen nach sich ziehen. Wenn die Sichten während des Umschreibens der Abfrage zusammengefügt werden, können diese Einschränkungen aufgehoben werden.
  - Beispiel - Umsetzungen von Unterabfragen in Verknüpfungen

    Wenn eine SELECT-Anweisung eine Unterabfrage enthält, kann die Auswahl der Reihenfolgeverarbeitung der Tabelle eingeschränkt sein.
  - Beispiel - Eliminierung überflüssiger Verknüpfungen

Während des Umschreibens der Abfrage können überflüssige Verknüpfungen entfernt werden, um die SELECT-Anweisung zu vereinfachen.

– Beispiel - Gemeinsame Spaltenberechnungen

  Wenn eine Abfrage verschiedene Funktionen verwendet, kann durch Umschreiben die Anzahl der erforderlichen Berechnungen reduziert werden.

- **Verschieben von Operationen**

  Zur Generierung der Abfrage mit der kleinstmöglichen Anzahl an Operationen und Vergleichselementen schreibt der Compiler Abfragen um, um Abfrageoperationen zu verschieben. Die folgenden Beispiele zeigen einige Operationen, die verschoben werden können:

  – Beispiel - Eliminierung von DISTINCT

    Während des Umschreibens einer Abfrage kann das Optimierungsprogramm den Zeitpunkt verschieben, zu dem die DISTINCT-Operation durchgeführt wird, um den Aufwand für diese Operation zu verringern. In dem bereitgestellten erweiterten Beispiel wird die DISTINCT-Operation vollständig entfernt.

  – Beispiel - Allgemeine Verschiebung von Vergleichselementen (Pushdown)

    Während des Umschreibens von Abfragen kann das Optimierungsprogramm die Reihenfolge der angewandten Vergleichselemente ändern, so dass die Vergleichselemente, die die Auswahl in größerem Maße einschränken, zum frühstmöglichen Zeitpunkt angewandt werden.

  – Beispiel - Dekorrelierung

    In einer Umgebung mit partitionierten Datenbanken erfordert das Verschieben von Ergebnismengen zwischen den Datenbankpartitionen hohen Aufwand. Den Umfang des Broadcastbetriebs (Rundsenden) an andere Datenbankpartitionen oder die Anzahl der Broadcastvorgänge zu reduzieren, oder beides, gehört zu den Zielen des Umschreibens von Abfragen.

- **Übersetzen von Vergleichselementen**

  Der SQL-Compiler schreibt Abfragen um, um vorhandene Vergleichselemente für eine bestimmte Abfrage in eine optimierte Form zu bringen. Die folgenden Beispiele zeigen einige der Vergleichselemente, die übersetzt werden können:

  – Beispiel - Hinzufügen implizierter Vergleichselemente

    Während des Umschreibens können Vergleichselemente der Abfrage hinzugefügt werden, um dem Optimierungsprogramm die Möglichkeit zu geben, weitere Tabellenverknüpfungen bei der Auswahl des günstigsten Zugriffsplans für die Abfrage in Betracht zu ziehen.

  – Beispiel - Transformationen von OR zu IN

    Während des Umschreibens einer Abfrage kann für einen effizienteren Zugriffsplan ein Vergleichselement OR in ein Vergleichselement IN übersetzt werden. Der SQL-Compiler kann auch ein Vergleichselement IN in ein Vergleichselement OR übersetzen, wenn diese Transformation einen günstigeren Zugriffsplan generieren würde.

Zugehörige Konzepte:

- „Beispiel für das Umschreiben durch den Compiler: Zusammenfügen von Sichten" auf Seite 166
- „Beispiel für das Umschreiben durch den Compiler: Eliminierung von DISTINCT" auf Seite 168
- „Beispiel für das Umschreiben durch den Compiler: implizierte Vergleichselemente" auf Seite 170
- „Spaltenkorrelation für mehrere Vergleichselemente" auf Seite 171

Beispiel für das Umschreiben durch den Compiler: Zusammenfügen von Sichten

Nehmen Sie zum Beispiel an, Sie haben Zugriff auf die beiden folgenden Sichten der Tabelle EMPLOYEE, von denen die eine die Mitarbeiter mit einer hohen Ausbildungsstufe (EDLEVEL) und die andere die Mitarbeiter mit einem Gehalt (SALARY) über 35.000 Dollar zeigt:

```
CREATE VIEW EMP_EDUCATION (EMPNO, FIRSTNME, LASTNAME, EDLEVEL) AS
SELECT EMPNO, FIRSTNME, LASTNAME, EDLEVEL
  FROM EMPLOYEE
 WHERE EDLEVEL > 17
CREATE VIEW EMP_SALARIES (EMPNO, FIRSTNAME, LASTNAME, SALARY) AS
SELECT EMPNO, FIRSTNME, LASTNAME, SALARY
  FROM EMPLOYEE
 WHERE SALARY > 35000
```

Jetzt wird beispielsweise die folgende Abfrage ausgeführt, um die Mitarbeiter, die eine hohe Ausbildungsstufe haben und deren Gehalt über 35.000 Dollar liegt, aufzulisten:

```
SELECT E1.EMPNO, E1.FIRSTNME, E1.LASTNAME, E1.EDLEVEL, E2.SALARY
  FROM EMP_EDUCATION E1,
       EMP_SALARIES  E2
 WHERE E1.EMPNO = E2.EMPNO
```

Während der Phase des Umschreibens könnten diese beiden Sichten zusammengefügt werden, um folgende Abfrage zu erstellen:

```
SELECT E1.EMPNO, E1.FIRSTNME, E1.LASTNAME, E1.EDLEVEL, E2.SALARY
  FROM EMPLOYEE E1,
       EMPLOYEE E2
 WHERE E1.EMPNO = E2.EMPNO
   AND E1.EDLEVEL > 17
   AND E2.SALARY  > 35000
```

Durch das Zusammenfügen der Anweisungen SELECT der beiden Sichten mit der vom Benutzer geschriebenen Anweisung SELECT kann das Optimierungsprogramm mehr Möglichkeiten bei der Auswahl des Zugriffsplans in Betracht ziehen. Darüber hinaus kann die Abfrage noch weiter umgeschrieben werden, wenn die beiden zusammengefügten Sichten dieselbe Basistabelle verwenden.

Beispiel - Umsetzungen von Unterabfragen in Verknüpfungen

Nehmen Sie an, eine Abfrage enthält die folgende Unterabfrage:

```
SELECT EMPNO, FIRSTNME, LASTNAME, PHONENO
FROM EMPLOYEE
WHERE WORKDEPT IN
      (SELECT DEPTNO
         FROM DEPARTMENT
        WHERE DEPTNAME = 'OPERATIONS')
```

Diese Unterabfrage wird vom SQL-Compiler in eine Verknüpfungsabfrage der folgenden Form umgewandelt:

```
SELECT DISTINCT EMPNO, FIRSTNME, LASTNAME, PHONENO
  FROM EMPLOYEE EMP,
       DEPARTMENT DEPT
 WHERE EMP.WORKDEPT = DEPT.DEPTNO
   AND DEPT.DEPTNAME = 'OPERATIONS'
```

Im Allgemeinen ist eine Verknüpfung in der Ausführung wesentlich effektiver als eine Unterabfrage.

Beispiel - Eliminierung überflüssiger Verknüpfungen

Manche geschriebenen oder generierten Abfragen enthalten unnötige Verknüpfungen. Abfragen wie die folgende könnten auch während des Umschreibens einer Abfrage generiert werden.

```
SELECT E1.EMPNO, E1.FIRSTNME, E1.LASTNAME, E1.EDLEVEL, E2.SALARY
  FROM EMPLOYEE E1,
       EMPLOYEE E2
 WHERE E1.EMPNO = E2.EMPNO
   AND E1.EDLEVEL > 17
   AND E2.SALARY  > 35000
```

In dieser Abfrage kann der SQL-Compiler die Verknüpfung eliminieren und die Abfrage auf folgende Form vereinfachen:

```
SELECT EMPNO, FIRSTNME, LASTNAME, EDLEVEL, SALARY
  FROM EMPLOYEE
 WHERE EDLEVEL > 17
   AND SALARY  > 35000
```

Im folgenden Beispiel wird davon ausgegangen, dass zwischen den Beispieltabellen EMPLOYEE und DEPARTMENT eine referenzielle Integritätsbedingung über die Abteilung (WORKDEPT/DEPNO) vorhanden ist. Zuerst wird eine Sicht erstellt.

```
CREATE VIEW PEPLVIEW
   AS SELECT FIRSTNME, LASTNAME, SALARY, DEPTNO, DEPTNAME, MGRNO
         FROM EMPLOYEE E DEPARTMENT D
         WHERE E.WORKDEPT = D.DEPTNO
```

Eine Abfrage wie die folgende:

```
SELECT LASTNAME, SALARY
   FROM PEPLVIEW
```

wird dann geändert in:

```
SELECT LASTNAME, SALARY
   FROM EMPLOYEE
   WHERE WORKDEPT NOT NULL
```

Beachten Sie bei dieser Situation, dass Benutzer die Abfrage eventuell nicht umschreiben können, selbst wenn sie wüssten, dass dies möglich ist, weil sie keinen Zugriff auf die zugrunde liegenden Tabellen haben. Sie haben eventuell nur Zugriff auf die oben gezeigte Sicht. Daher muss diese Art von Optimierung im Datenbankmanager ausgeführt werden.

Redundanz in Verknüpfungen mit referenzieller Integrität ist in folgenden Fällen wahrscheinlich:

- Sichten sind mit Verknüpfungen definiert.
- Abfragen werden automatisch generiert.

  Es gibt zum Beispiel automatisierte Tools in Abfragemanagern, die das Schreiben optimierter Abfragen durch Benutzer verhindern.

Beispiel - Gemeinsame Spaltenberechnungen

Bei Verwendung mehrerer Funktionen in einer Abfrage können zahlreiche Berechnungen entstehen, die zeitintensiv sind. Durch Reduzieren der für die Abfrage erforderlichen Anzahl von Berechnungen kann der Zugriffsplan verbessert werden. Eine Abfrage, die mehrere Funktionen verwendet, wie zum Beispiel:

```
SELECT SUM(SALARY+BONUS+COMM) AS OSUM,
  AVG(SALARY+BONUS+COMM) AS OAVG,
  COUNT(*) AS OCOUNT
FROM EMPLOYEE;
```

wird vom SQL-Compiler wie folgt umgewandelt:

```
SELECT OSUM,
  OSUM/OCOUNT
  OCOUNT
FROM (SELECT SUM(SALARY+BONUS+COMM) AS OSUM,
  COUNT(*) AS OCOUNT
FROM EMPLOYEE) AS SHARED_AGG;
```

Durch dieses Umschreiben benötigt die Abfrage statt 2 Summen und 2 Zählern nur noch 1 Summe und 1 Zähler.

Zugehörige Konzepte:

- „Der SQL-Compilerprozess" auf Seite 157
- „Methoden zum Umschreiben von Abfragen und Beispiele" auf Seite 164

Beispiel für das Umschreiben durch den Compiler: Eliminierung von DISTINCT

Für das folgende Abfragebeispiel wird angenommen, dass die Spalte EMPNO als Primärschlüssel der Tabelle EMPLOYEE definiert wurde:

```
SELECT DISTINCT EMPNO, FIRSTNME, LASTNAME
  FROM EMPLOYEE
```

Diese Abfrage würde so umgeschrieben, dass die Klausel DISTINCT entfernt wird:

```
SELECT EMPNO, FIRSTNME, LASTNAME
  FROM EMPLOYEE
```

Da hier der Primärschlüssel ausgewählt wird, weiß der SQL-Compiler, dass jede zurückgelieferte Zeile bereits eindeutig ist. In diesem Fall wird das Schlüsselwort DISTINCT nicht benötigt. Wenn die Abfrage nicht umgeschrieben wird, erstellt das Optimierungsprogramm einen Plan, der die nötigen Verarbeitungsschritte, wie zum Beispiel eine Sortierung, enthält, um sicherzustellen, dass die Spalten eindeutig sind.

Beispiel - Allgemeines Verschieben von Vergleichselementen (Pushdown)

Durch das Ändern der Ebene, auf der Vergleichselemente normalerweise angewendet werden, lässt sich eventuell eine Leistungsverbesserung erreichen. Zum Beispiel sei die folgende Sicht gegeben, die eine Liste aller Mitarbeiter der Abteilung „D11" zusammenstellt:

```
CREATE VIEW D11_EMPLOYEE
 (EMPNO, FIRSTNME, LASTNAME, PHONENO, SALARY, BONUS, COMM)
AS SELECT EMPNO, FIRSTNME, LASTNAME, PHONENO, SALARY, BONUS, COMM
     FROM EMPLOYEE
    WHERE WORKDEPT = 'D11'
```

Es wird nun die folgende Abfrage aufgesetzt:

```
SELECT FIRSTNME, PHONENO
  FROM D11_EMPLOYEE
 WHERE LASTNAME = 'BROWN'
```

In der Phase des Umschreibens verschiebt der Compiler das Vergleichselement LASTNAME = 'BROWN' nach unten in die Sicht D11_EMPLOYEE. Dadurch kann das Vergleichselement früher und möglicherweise effektiver angewandt werden. Die tatsächliche Abfrage, die in diesem Beispiel zur Ausführung kommen könnte, sieht folgendermaßen aus:

```
SELECT FIRSTNME, PHONENO
  FROM EMPLOYEE
 WHERE LASTNAME = 'BROWN'
   AND WORKDEPT = 'D11'
```

Das Verschieben von Vergleichselementen ist nicht auf Sichten beschränkt. Beispiele für andere Situationen, in denen Vergleichselemente verschoben werden können, sind Klauseln UNION, GROUP BY und abgeleitete Tabellen (verschachtelte Tabellenausdrücke oder allgemeine Tabellenausdrücke).

Beispiel - Dekorrelierung

In einer Umgebung mit partitionierten Datenbanken kann der SQL-Compiler die folgende Abfrage umschreiben:

Lokalisieren aller Mitarbeiter, die in der Programmierung arbeiten und unterbezahlt sind.

```
SELECT P.PROJNO, E.EMPNO, E.LASTNAME, E.FIRSTNAME,
       E.SALARY+E.BONUS+E.COMM AS COMPENSATION
  FROM EMPLOYEE E, PROJECT P
 WHERE P.EMPNO = E.EMPNO
   AND P.PROJNAME LIKE '%PROGRAMMING%'
   AND E.SALARY+E.BONUS+E.COMM <
     (SELECT AVG(E1.SALARY+E1.BONUS+E1.COMM)
        FROM EMPLOYEE E1, PROJECT P1
        WHERE P1.PROJNAME LIKE '%PROGRAMMING%'
          AND P1.PROJNO = A.PROJNO
          AND E1.EMPNO = P1.EMPNO)
```

Da die Abfrage korreliert ist und wahrscheinlich weder PROJECT noch EMPLOYEE über die Spalte PROJNO partitioniert sind, wird möglicherweise jedes Projekt im Broadcastbetrieb an jede Datenbankpartition übermittelt. Außerdem müsste die Unterabfrage viele Male ausgewertet werden.

Der SQL-Compiler kann die Abfrage wie folgt umschreiben:

- Die eindeutige Liste (DISTINCT) der Mitarbeiter (Employees) ermitteln, die an Programmierungsprojekten arbeiten, und diese DIST_PROJS nennen. Diese Liste muss mit DISTINCT erstellt werden, damit sichergestellt wird, dass die Spaltenberechnung nur einmal pro Projekt erfolgt:

  ```
  WITH DIST_PROJS(PROJNO, EMPNO) AS
  (SELECT DISTINCT PROJNO, EMPNO
   FROM PROJECT P1
   WHERE P1.PROJNAME LIKE '%PROGRAMMING%')
  ```

- Die eindeutige Liste der Mitarbeiter, die an Programmierungsprojekten arbeiten mit der Mitarbeitertabelle verknüpfen, um die durchschnittliche Entlohnung je Projekt (AVG_PER_PROJ) zu ermitteln:

```
AVG_PER_PROJ(PROJNO, AVG_COMP) AS
(SELECT P2.PROJNO, AVG(E1.SALARY+E1.BONUS+E1.COMM)
 FROM EMPLOYEE E1, DIST_PROJS P2
 WHERE E1.EMPNO = P2.EMPNO
 GROUP BY P2.PROJNO)
```

- Die neue Abfrage würde wie folgt aussehen:

```
SELECT P.PROJNO, E.EMPNO, E.LASTNAME, E.FIRSTNAME,
       E.SALARY+E.BONUS+E.COMM AS COMPENSATION
 FROM PROJECT P, EMPLOYEE E, AVG_PER_PROG A
WHERE P.EMPNO = E.EMPNO
  AND P.PROJNAME LIKE '%PROGRAMMING%'
  AND P.PROJNO = A.PROJNO
  AND E.SALARY+E.BONUS+E.COMM < A.AVG_COMP
```

Die umgeschriebene SQL-Abfrage berechnet AVG_COMP je Projekt (AVG_PRE_PROJ) und kann das Ergebnis an alle Datenbankpartitionen übermitteln, in denen die Tabelle EMPLOYEE enthalten ist.

Zugehörige Konzepte:

- „Der SQL-Compilerprozess" auf Seite 157
- „Methoden zum Umschreiben von Abfragen und Beispiele" auf Seite 164

Beispiel für das Umschreiben durch den Compiler: implizierte Vergleichselemente

Die folgende Abfrage erstellt eine Liste der Manager, deren Abteilung an Abteilung „E01" berichten, und der Projekte, für die die Manager verantwortlich sind:

```
SELECT DEPT.DEPTNAME DEPT.MGRNO, EMP.LASTNAME, PROJ.PROJNAME
  FROM DEPARTMENT DEPT,
       EMPLOYEE   EMP,
       PROJECT    PROJ
 WHERE DEPT.ADMRDEPT = 'E01'
   AND DEPT.MGRNO = EMP.EMPNO
   AND EMP.EMPNO  = PROJ.RESPEMP
```

Beim Umschreiben der Abfrage wird das folgende implizierte Vergleichselement hinzugefügt:

```
DEPT.MGRNO = PROJ.RESPEMP
```

Als Ergebnis dieses Umschreibens kann das Optimierungsprogramm weitere Verknüpfungen in die Berechnung mit einbeziehen, wenn es versucht, den günstigsten Zugriffsplan für die Abfrage auszuwählen.

Neben der oben gezeigten Ausnutzung der Transitivität der Vergleichselemente werden durch das Umschreiben auch zusätzliche lokale Vergleichselemente aufgrund der durch Gleichheitsvergleichselemente implizierten Transitivität abgeleitet. Zum Beispiel stellt die folgende Abfrage eine Liste der Namen der Abteilungen (deren Abteilungsnummer größer als „E00" ist) und der Mitarbeiter, die in diesen Abteilungen arbeiten, zusammen.

```
SELECT EMPNO, LASTNAME, FIRSTNAME, DEPTNO, DEPTNAME
  FROM EMPLOYEE EMP,
       DEPARTMENT DEPT
 WHERE EMP.WORKDEPT = DEPT.DEPTNO
   AND DEPT.DEPTNO > 'E00'
```

Dieser Abfrage wird in der Phase des Umschreibens das folgende implizierte Vergleichselement hinzugefügt:

```
EMP.WORKDEPT > 'E00'
```

Das Ergebnis dieses Umschreibens besteht darin, dass das Optimierungsprogramm die Anzahl der Zeilen, die verknüpft werden müssen, verringern kann.

Beispiel - Transformationen von OR zu IN

Nehmen Sie an, eine Klausel OR verknüpft zwei oder mehr einfache Gleichheitsvergleichselemente für dieselbe Spalte, wie im folgenden Beispiel:

```
SELECT *
  FROM EMPLOYEE
 WHERE DEPTNO = 'D11'
    OR DEPTNO = 'D21'
    OR DEPTNO = 'E21'
```

Wenn es für die Spalte DEPTNO keinen Index gibt, erlaubt die Umwandlung der Klausel OR in das folgende Vergleichselement IN eine effektivere Verarbeitung der Abfrage:

```
SELECT *
  FROM EMPLOYEE
 WHERE DEPTNO IN ('D11', 'D21', 'E21')
```

Anmerkung: In einigen Fällen kann der Datenbankmanager ein IN-Vergleichselement in eine Gruppe von Klauseln OR umwandeln, so dass logische OR-Verknüpfungen für Indizes (Index ORing) durchgeführt werden können.

Zugehörige Konzepte:

- „Der SQL-Compilerprozess" auf Seite 157
- „Methoden zum Umschreiben von Abfragen und Beispiele" auf Seite 164

Spaltenkorrelation für mehrere Vergleichselemente

Ihre Anwendungen enthalten möglicherweise Abfragen, die mit Verknüpfungen so konstruiert sind, dass zwei Tabellen über mehr als ein Verknüpfungsvergleichselement verknüpft werden. Dies ist nicht ungewöhnlich, wenn Abfragen die Beziehungen zwischen ähnlichen zusammengehörigen Spalten in verschiedenen Tabellen bestimmen müssen.

Betrachten Sie zum Beispiel einen Hersteller, der Produkte aus Rohmaterialien verschiedener Farben, Elastizitäten und Qualitäten produziert. Das fertige Produkt hat dieselbe Farbe und Elastizität wie das Rohmaterial, aus dem es hergestellt ist. Der Hersteller führt die folgende Abfrage aus:

```
SELECT PRODUCT.NAME, RAWMATERIAL.QUALITY FROM PRODUCT, RAWMATERIAL
   WHERE PRODUCT.COLOR       =  RAWMATERIAL.COLOR
     AND PRODUCT.ELASTICITY  =  RAWMATERIAL.ELASTICITY
```

Diese Abfrage liefert die Namen und die Qualität der Rohmaterialien aller Produkte. Zwei Vergleichselemente werden für die Verknüpfung verwendet:

```
PRODUCT.COLOR       =  RAWMATERIAL.COLOR
PRODUCT.ELASTICITY  =  RAWMATERIAL.ELASTICITY
```

Wenn das Optimierungsprogramm einen Plan zur Ausführung dieser Abfrage auswählt, berechnet es die Selektivität jedes der beiden Vergleichselemente. Dabei geht es davon aus, dass sie unabhängig sind, das heißt, dass alle Variationen von Elastizität für jede Farbe vorkommen und umgekehrt für jede Elastizitätsstufe Rohmate-

rial in jeder Farbe verfügbar ist. Dann schätzt es die Gesamtselektivität des Vergleichselementpaares ab, indem es auf Katalogstatistikdaten für jede Tabelle über die Anzahl der Elastizitätsstufen und die Anzahl verschiedener Farben zurückgreift. Ausgehend von diesem Schätzwert kann es beispielsweise eine Verknüpfung über Verschachtelungsschleife (Nested Loop Join) einer Mischverknüpfung (Merge Join) vorziehen oder umgekehrt.

Es kann jedoch vorkommen, dass diese beiden Vergleichselemente nicht unabhängig sind. Zum Beispiel könnte es der Fall sein, dass die hochelastischen Materialien nur in wenigen Farben verfügbar sind und die sehr wenig elastischen Materialien nur in einigen anderen Farben verfügbar sind, die sich von den Farben der elastischen Materialien unterscheiden. In diesem Fall eliminiert die kombinierte Selektivität der Vergleichselemente weniger Zeilen, so dass die Abfrage mehr Zeilen liefert. Betrachten Sie den extremen Fall, dass es nur eine Stufe von Elastizität für jede Farbe gibt und umgekehrt. Hier könnte jedes der beiden Vergleichselemente logisch ganz weggelassen werden, da es vom jeweils anderen impliziert wird. Das Optimierungsprogramm wählt vielleicht nicht mehr den besten Plan. Zum Beispiel könnte es einen Plan mit Verknüpfung über Verschachtelungsschleife wählen, obwohl eine Mischverknüpfung schneller wäre.

Bei anderen Datenbankprodukten haben Datenbankadministratoren versucht, dieses Leistungsproblem dadurch in den Griff zu bekommen, dass sie Statistiken im Katalog aktualisierten, um eines der Vergleichselemente weniger selektiv erscheinen zu lassen, jedoch kann dieses Vorgehen zu unerwünschten Nebeneffekten bei anderen Abfragen führen.

Das Optimierungsprogramm von DB2® UDB versucht, eine Korrelation von Verknüpfungsvergleichselementen zu entdecken und zu kompensieren, wenn Sie einen Index für diese Spalten definieren oder wenn Sie Gruppenspaltenstatistiken für die entsprechenden Spalten erfassen und pflegen.

In dem obigen Beispiel mit Farben und Elastizitäten von Materialien könnten Sie einen eindeutigen Index definieren, der folgende Spaltenpaare abdeckt. Entweder:

```
PRODUCT.COLOR, PRODUCT.ELASTICITY
```

oder

```
RAWMATERIAL.COLOR, RAWMATERIAL.ELASTICITY
```

oder beide.

Damit das Optimierungsprogramm die Korrelation erkennen kann, dürfen die Nicht-INCLUDE-Spalten dieses Index nur die korrelierten Spalten sein. Der Index kann darüber hinaus auch INCLUDE-Spalten enthalten, um Abfragen nur über Indexsuchen erfüllen zu können. Wenn es mehr als zwei korrelierte Spalten in Verknüpfungsvergleichselementen gibt, stellen Sie sicher, dass Sie einen eindeutigen Index definieren, der alle diese Spalten abdeckt. In vielen Fällen stellen korrelierte Spalten innerhalb einer Tabelle den Primärschlüssel der Tabelle dar. Da ein Primärschlüssel immer eindeutig ist, brauchen Sie keinen weiteren eindeutigen Index zu definieren.

Stellen Sie nach der Erstellung der entsprechenden Indizes sicher, dass die Tabellenstatistiken aktuell sind und nicht aus irgendeinem Grund, zum Beispiel um das Optimierungsprogramm zu beeinflussen, von den wahren Werten abweichend manuell geändert wurden.

Das Optimierungsprogramm verwendet die Informationen in den Spalten FIRSTnKEYCARD und FULLKEYCARD der Statistiktabellen für eindeutige Indizes, um Fälle von Korrelation zu erkennen und die errechneten kombinierten Selektivitäten der korrelierten Vergleichselemente dynamisch anzupassen, um so eine realistischere Schätzung der Größe und des Aufwands der Verknüpfung zu erhalten.

Alternativ können Spaltengruppenstatistiken für eine Gruppe von Spalten gesammelt werden. Im obigen Elastizitätsbeispiel könnten Sie Statistiken für die Spalten PRODUCT.COLOR, PRODUCT.ELASTICITY und/oder RAWMATERIAL.COLOR, RAWMATERIAL.ELASTCITY sammeln.

Spaltengruppenstatistiken werden mit der Option ON COLUMNS des Dienstprogramms RUNSTATS erfasst. Wenn Sie zum Beispiel die Spaltengruppenstatistiken für PRODUCT.COLOR und PRODUCT.ELASTICITY sammeln wollen, führen Sie den folgenden RUNSTATS-Befehl aus:

```
RUNSTATS ON TABLE product ON COLUMNS ((color, elasticity))
```

Korrelation einfacher Gleichheitsvergleichselemente

Zusätzlich zur Korrelation von Verknüpfungsvergleichselementen verwaltet das Optimierungsprogramm auch die Korrelation mit einfachen Gleichheitsvergleichselementen des Typs SPALTE = *konstante*. Betrachten Sie zum Beispiel eine Tabelle mit verschiedenen Typen von Fahrzeugen vor, die jeweils mit MARKE (d. h. Hersteller), MODELL, JAHR, FARBE und AUSFÜHRUNG (z. B. Limousine, Kombi, Sport- oder Nutzfahrzeug) eingetragen sind. Da die weitaus meisten Hersteller die gleichen Standardfarben für jedes ihrer Modelle und jede ihrer Ausführungen Jahr für Jahr liefern, sind die Vergleichselemente für die Spalte FARBE wahrscheinlich von denen für die Spalten MARKE, MODELL, AUSFÜHRUNG oder JAHR unabhängig. Die Vergleichselemente MARKE und MODELL sind jedoch in keinem Fall unabhängig voneinander, da nur jeweils ein Fahrzeughersteller ein Modell mit einem bestimmten Namen produziert. Identische Modellnamen, die von zwei oder mehr Fahrzeugherstellern verwendet werden, sind sehr unwahrscheinlich und mit Sicherheit von den Fahrzeugherstellern nicht gewollt.

Wenn für die beiden Spalten MARKE und MODELL ein Index vorhanden ist oder Spaltengruppenstatistiken erfasst sind, verwendet das Optimierungsprogramm die Statistikdaten über diesen Index oder die Spalten, um die kombinierte Zahl an unterschiedlichen Werten festzustellen und die Selektivitäts- oder Kardinalitätsschätzung für die Korrelation zwischen diesen beiden Spalten anzupassen. Wenn solche Vergleichselemente nicht zur Verknüpfung verwendet werden, benötigt das Optimierungsprogramm keinen eindeutigen Index, um die Anpassung vorzunehmen.

Zugehörige Konzepte:

- „Der SQL-Compilerprozess" auf Seite 157
- „Methoden zum Umschreiben von Abfragen und Beispiele" auf Seite 164

Abfrageoptimierung mit der Bindeoption REOPT

Damit eine Abfrageoptimierung (bzw. Reoptimierung) für statische und dynamische SQL-Anweisungen, die Hostvariablen, Sonderregister oder Parametermarken enthalten, ausgeführt werden kann, binden Sie das Paket mit der Bindeoption REOPT. Wenn diese Option verwendet wird, wird der Zugriffspfad für eine SQL-Anweisung, die sowohl zu dem Paket gehört als auch Hostvariablen, Parametermarken oder Sonderregister enthält, mit den Werten dieser Variablen, und nicht

mit Standardschätzwerten, die der Compiler auswählt, optimiert. Die Optimierung erfolgt bei der Ausführung der Abfrage, wenn die Werte verfügbar sind.

Zugehörige Konzepte:

- „Effects of REOPT on static SQL" in *Application Development Guide: Programming Client Applications*
- „Effects of REOPT on dynamic SQL" in *Application Development Guide: Programming Client Applications*

Datenzugriffsmethoden

Dieser Abschnitt beschreibt die Methoden, die das Optimierungsprogramm auswählen kann, um auf die von einer Abfrage angeforderten Daten zuzugreifen.

Datenzugriffsmethoden

Bei der Kompilierung einer SQL-Anweisung schätzt das Optimierungsprogramm den Ausführungsaufwand der verschiedenen Methoden ab, die die Anforderung erfüllen würden. Auf der Grundlage dieser Schätzungen wählt das Optimierungsprogramm einen optimalen Zugriffsplan aus. Ein *Zugriffsplan* gibt die Reihenfolge von Operationen an, die erforderlich sind, um eine SQL-Anweisung auszuführen. Wenn ein Anwendungsprogramm gebunden wird, wird ein *Paket* erstellt. Dieses Paket enthält Zugriffspläne für alle statischen SQL-Anweisungen in dem entsprechenden Anwendungsprogramm. Die Zugriffspläne für dynamische SQL-Anweisungen werden zum Zeitpunkt der Ausführung der Anwendung erstellt.

Es gibt zwei Methoden für den Zugriff auf Daten in einer Tabelle:

- Sequenzielles Durchsuchen der gesamten Tabelle
- Lokalisieren bestimmter Tabellenzeilen durch einen vorherigen Zugriff auf einen Index für die Tabelle

Zur Erstellung der von der Abfrage angeforderten Ergebnisse werden Zeilen in Abhängigkeit von den Bedingungen des Vergleichselements ausgewählt, das in der Regel in einer WHERE-Klausel angegeben wird. Die ausgewählten Zeilen in den Tabellen, auf die zugegriffen wird, werden verknüpft, um die Ergebnismenge zu erstellen, und die Ergebnismenge wird unter Umständen weiter verarbeitet, indem die Ausgabe gruppiert oder sortiert wird.

Zugehörige Konzepte:

- „Der SQL-Compilerprozess" auf Seite 157
- „Datenzugriff über Indexsuchen" auf Seite 174
- „Arten des Indexzugriffs" auf Seite 178
- „Indexzugriff und Clusterverhältnisse" auf Seite 180

Datenzugriff über Indexsuchen

Als *Indexsuche* wird der Vorgang bezeichnet, bei dem der Datenbankmanager zu folgenden Zwecken auf einen Index zugreift:

- Eingrenzen der Menge der den Bedingungen entsprechenden Zeilen (durch Durchsuchen der Zeilen in einem bestimmten Bereich des Index) vor dem Zugriff auf die Basistabelle. Der *Suchbereich* des Index (der Start- und der Stoppunkt der Suche) wird durch die Werte in der Abfrage festgelegt, mit denen Indexspalten verglichen werden.

- Sortieren der Ausgabe.
- Direktes Abrufen der angeforderten Spaltendaten. Wenn sich alle angeforderten Daten im Index befinden, braucht kein Zugriff auf die Basistabelle stattzufinden. Dies wird als *reiner Indexzugriff* bezeichnet.

Wenn Indizes mit der Option ALLOW REVERSE SCANS erstellt wurden, können Indexsuchen auch in entgegengesetzter Richtung zu der Richtung durchgeführt werden, in der die Indizes definiert wurden.

Anmerkung: Das Optimierungsprogramm wählt eine Tabellensuche, wenn kein entsprechender Index erstellt wurde oder eine Indexsuche aufwendiger wäre. Eine Indexsuche kann aufwendiger sein, wenn die Tabelle klein ist, das Indexclusterverhältnis niedrig ist oder die Abfrage die meisten der Tabellenzeilen anfordert. Ob der Zugriffsplan eine Tabellensuche oder eine Indexsuche verwendet, lässt sich mit Hilfe der SQL-EXPLAIN-Einrichtung feststellen.

Indexsuchen zur Begrenzung eines Bereichs

Zur Bestimmung, ob ein Index für eine bestimmte Abfrage verwendet werden kann, wertet das Optimierungsprogramm jede Spalte des Index beginnend bei der ersten Spalte aus, um zu überprüfen, ob sie zur Erfüllung von Gleichheitsvergleichselementen und anderen Vergleichselementen in der WHERE-Klausel verwendet werden kann. Ein *Vergleichselement* ist ein Element einer Suchbedingung in einer Klausel WHERE, das eine Vergleichsoperation definiert oder impliziert. Vergleichselemente können zur Begrenzung des Bereichs einer Indexsuche zu folgenden Zwecken verwendet werden:

- Testen auf Gleichheit mit einer Konstante, einer Hostvariablen, einem Ausdruck, der zu einer Konstanten ausgewertet wird, oder einem Schlüsselwort.
- Testen auf „IS NULL" oder „IS NOT NULL".
- Testen auf Gleichheit mit einer einfachen Unterabfrage, d. h. mit einer Unterabfrage, die weder ANY, ALL oder SOME noch einen Verweis über eine korrelierte Spalte auf den unmittelbar übergeordneten Abfrageblock (d. h. auf die SELECT-Anweisung, für die diese Unterabfrage eine Unterauswahl ist) enthält.
- Testen auf strenge oder einschließende Ungleichheit.

Die folgenden Beispiele veranschaulichen, wann ein Index zur Begrenzung eines Bereichs verwendet werden könnte:

- Betrachten Sie einen Index mit der folgenden Definition:

```
INDEX IX1:  NAME    ASC,
            DEPT    ASC,
            MGR     DESC,
            SALARY  DESC,
            YEARS   ASC
```

  In diesem Fall könnten die folgenden Vergleichselemente zur Begrenzung des Bereichs bei der Suche im Index IX1 verwendet werden:

```
WHERE  NAME   = :hv1
  AND  DEPT = :hv2
```

  oder

```
WHERE  MGR  = :hv1
  AND  NAME = :hv2
  AND  DEPT = :hv3
```

  Beachten Sie, dass in der zweiten WHERE-Klausel die Vergleichselemente nicht in derselben Reihenfolge angegeben werden müssen, in der die Schlüsselspalten

im Index erscheinen. In den Beispielen werden zwar Hostvariablen (hv = Hostvariable) verwendet, jedoch hätten andere Variablen wie Parametermarken, Ausdrücke oder Konstanten dieselbe Wirkung.

- Betrachten Sie einen einzelnen Index, der mit dem Parameter ALLOW REVERSE SCANS erstellt wurde. Solche Indizes unterstützen Suchoperationen in der Richtung, die bei der Erstellung des Index definiert wurde, und Suchoperationen in entgegengesetzter oder umgekehrter Richtung. Die Anweisung könnte etwa wie folgt aussehen:

  ```
  CREATE INDEX iname ON tname (cname DESC) ALLOW REVERSE SCANS
  ```

  In diesem Fall wird der Index (iname) nach den absteigenden (DESCending) Werten der Spalte cname gebildet. Durch Zulassen von umgekehrten Suchoperationen kann eine Suche in aufsteigender Folge durchgeführt werden, obwohl der Index für die Spalte für Suchoperationen in absteigender Folge definiert ist. Die tatsächliche Verwendung des Index in beiden Richtungen wird nicht von Ihnen, sondern vom Optimierungsprogramm bei der Erstellung und Auswahl von Zugriffsplänen gesteuert.

In der folgenden Klausel WHERE würden nur die Vergleichselemente für NAME und DEPT verwendet, um den Bereich der Indexsuche einzugrenzen, jedoch nicht die Vergleichselemente für SALARY und YEARS:

```
WHERE  NAME   = :hv1
  AND  DEPT   = :hv2
  AND  SALARY = :hv4
  AND  YEARS  = :hv5
```

Der Grund dafür ist der, dass es eine Schlüsselspalte (MGR) gibt, die diese Spalten von den ersten beiden Indexschlüsselspalten trennt, so dass die Reihenfolge nicht gewährleistet wäre. Wenn aber der Bereich einmal durch die Vergleichselemente NAME = :hv1 und DEPT = :hv2 festgelegt ist, können die verbleibenden Vergleichselemente an den verbleibenden Indexschlüsselspalten ausgewertet werden.

Indexsuchen zum Testen von Ungleichheit

Bestimmte Ungleichheitsvergleichselemente können den Bereich einer Indexsuche begrenzen. Es gibt zwei Typen von Ungleichheitsvergleichselementen:

- Vergleichselemente für strenge Ungleichheit

  Die Operatoren für strenge (ausschließende) Ungleichheit, die für bereichsbegrenzende Vergleichselemente verwendet werden, sind > (größer als) und < (kleiner als).

  Nur eine Spalte mit Vergleichselementen für strenge Ungleichheit wird zur Begrenzung des Bereichs für eine Indexsuche berücksichtigt. Im folgenden Beispiel können die Vergleichselemente für die Spalten NAME und DEPT verwendet werden, um den Bereich einzugrenzen, aber das Vergleichselement für die Spalte MGR nicht.

  ```
  WHERE  NAME   = :hv1
    AND  DEPT   > :hv2
    AND  DEPT   < :hv3
    AND  MGR    < :hv4
  ```

- Vergleichselemente für einschließende Ungleichheit

  Die folgenden Operatoren für einschließende Ungleichheit können für Vergleichselemente verwendet werden, die einen Bereich begrenzen:

  – >= und <=

– BETWEEN
– LIKE

Zur Begrenzung eines Bereichs für eine Indexsuche werden mehrere Spalten mit Vergleichselementen einschließender Ungleichheit berücksichtigt. Im folgenden Beispiel können alle Vergleichselemente zur Eingrenzung des Bereichs der Indexsuche verwendet werden:

```
WHERE  NAME   = :hv1
  AND  DEPT  >= :hv2
  AND  DEPT  <= :hv3
  AND  MGR   <= :hv4
```

Zur weiteren Verdeutlichung dieses Beispiels seien die folgenden Werte angenommen: `:hv2 = 404`, `:hv3 = 406` und `:hv4 = 12345`. Der Datenbankmanager durchsucht den Index für die Abteilungen (DEPT) 404 und 405 ganz, bricht aber die Suche in Abteilung 406 ab, wenn er auf den ersten Manager trifft, der eine Personalnummer (Spalte MGR) über dem Wert 12345 hat.

Indexsuchen zum Sortieren von Daten

Wenn die Abfrage eine sortierte Ausgabe erfordert, kann ein Index zum Sortieren der Daten verwendet werden, wenn die ordnenden Spalten nacheinander, angefangen bei der ersten Indexschlüsselspalte, im Index vertreten sind. Ordnen oder Sortieren kann das Ergebnis solcher Operationen wie ORDER BY, DISTINCT, GROUP BY, Unterabfrage mit „= ANY", Unterabfrage mit „> ALL", Unterabfrage mit „< ALL", INTERSECT bzw. EXCEPT sowie UNION sein. Eine Ausnahme ist der Fall, wenn die Indexschlüsselspalten auf Gleichheit mit „konstanten Werten" verglichen werden, das heißt mit einem beliebigen Ausdruck, der zu einer Konstanten ausgewertet wird. In diesem Fall können die ordnenden Spalten andere als die ersten Indexschlüsselspalten sein.

Betrachten Sie die folgende Abfrage:

```
WHERE NAME = 'JONES'
  AND DEPT = 'D93'
ORDER BY MGR
```

Für diese Abfrage könnte der Index verwendet werden, um die Zeilen zu sortieren, da die Spalten NAME und DEPT immer dieselben Werte haben und so geordnet sind. Das heißt, die gezeigten Klauseln WHERE und ORDER BY sind äquivalent mit folgenden Klauseln:

```
WHERE NAME = 'JONES'
  AND DEPT = 'D93'
ORDER BY NAME, DEPT, MGR
```

Ein eindeutiger Index kann auch zum Verkürzen einer Sortieranforderung verwendet werden. Betrachten Sie die folgende Indexdefinition und die ORDER BY-Klausel:

```
UNIQUE INDEX IX0:  PROJNO  ASC
SELECT PROJNO, PROJNAME, DEPTNO
  FROM PROJECT
ORDER BY PROJNO, PROJNAME
```

Eine zusätzliche Sortierung über die Spalte PROJNAME ist nicht erforderlich, da der Index IX0 bereits sicherstellt, dass die Werte der Spalte PROJNO eindeutig sind. Diese Eindeutigkeit sorgt dafür, dass es nur einen Wert für PROJNAME für jeden Wert von PROJNO gibt.

Zugehörige Konzepte:
- „Datenzugriffsmethoden" auf Seite 174
- „Indexstruktur" auf Seite 27
- „Arten des Indexzugriffs" auf Seite 178
- „Indexzugriff und Clusterverhältnisse" auf Seite 180

Arten des Indexzugriffs

In einigen Fällen kann das Optimierungsprogramm feststellen, dass alle Daten, die eine Abfrage aus einer Tabelle anfordert, aus einem Index für die Tabelle abgerufen werden können. In anderen Fällen kann das Optimierungsprogramm auch mehrere Indizes für den Zugriff auf Tabellen verwenden. Im Fall von Bereichsclustertabellen kann der Zugriff auf Daten über einen „virtuellen" Index erfolgen, der die Positionen von Datensätzen berechnet.

Reiner Indexzugriff

In einigen Fällen können alle angeforderten Daten aus dem Index abgerufen werden, ohne auf die Tabelle zuzugreifen. Dies wird als *reiner Indexzugriff* bezeichnet.

Betrachten Sie die folgende Indexdefinition zur Illustration des reinen Indexzugriffs:

```
INDEX IX1:  NAME    ASC,
            DEPT    ASC,
            MGR     DESC,
            SALARY  DESC,
            YEARS   ASC
```

Die folgende Abfrage kann nur durch den Zugriff auf den Index ohne Lesen der Basistabelle erfüllt werden:

```
SELECT NAME, DEPT, MGR, SALARY
  FROM EMPLOYEE
 WHERE NAME = 'SMITH'
```

Häufig sind erforderliche Spalten jedoch nicht im Index vertreten. Um die Daten für diese Spalten abzurufen, müssen Tabellenzeilen gelesen werden. Damit das Optimierungsprogramm die Möglichkeit hat, einen reinen Indexzugriff zu wählen, erstellen Sie einen eindeutigen Index mit INCLUDE-Spalten. Betrachten Sie zum Beispiel die folgende Indexdefinition:

```
CREATE UNIQUE INDEX IX1 ON EMPLOYEE
   (NAME ASC)
    INCLUDE (DEPT, MGR, SALARY, YEARS)
```

Dieser Index gewährleistet die Eindeutigkeit der Datenwerte in der Spalte NAME und speichert und pflegt darüber hinaus Daten für die Spalten DEPT, MGR, SALARY und YEARS. Dadurch kann die folgende Abfrage nur durch einen Zugriff auf den Index erfüllt werden:

```
SELECT NAME, DEPT, MGR, SALARY
   FROM EMPLOYEE
   WHERE NAME='SMITH'
```

Wenn Sie das Hinzufügen von INCLUDE-Spalten für einen Index in Betracht ziehen, müssen Sie jedoch abwägen, ob der zusätzliche Speicherplatzbedarf und der Pflegeaufwand gerechtfertigt sind. Wenn Abfragen, die durch einen solchen Index erfüllt werden können, nur selten ausgeführt werden, ist der Zusatzaufwand möglicherweise nicht gerechtfertigt.

Zugriff auf mehrere Indizes

Das Optimierungsprogramm kann entscheiden, mehrere Indizes für dieselbe Tabelle zu durchsuchen, um die Vergleichselemente einer WHERE-Klausel zu erfüllen. Betrachten Sie zum Beispiel die beiden folgenden Indexdefinitionen:

```
INDEX IX2:  DEPT    ASC
INDEX IX3:  JOB     ASC,
            YEARS   ASC
```

Die folgenden Vergleichselemente können durch die Verwendung der beiden Indizes erfüllt werden:

```
WHERE DEPT = :hv1
  OR (JOB   = :hv2
  AND YEARS >= :hv3)
```

Das Durchsuchen des Index IX2 liefert eine Liste von Zeilen-IDs (RIDs), die das Vergleichselement `DEPT = :hv1` erfüllen. Das Durchsuchen des Index IX3 liefert eine Liste der RIDs, die das Vergleichselement `JOB = :hv2 AND YEARS >= :hv3` erfüllen. Diese beiden Listen von RIDs werden kombiniert und doppelte Werte entfernt, bevor auf die Tabelle zugegriffen wird. Diese Methode wird als *logische ODER-Verknüpfung von Indizes* (Index ORing) bezeichnet.

Die OR-Verknüpfung von Indizes kann auch für Vergleichselemente mit der Klausel IN wie in folgendem Beispiel verwendet werden:

```
WHERE DEPT IN (:hv1, :hv2, :hv3)
```

Während der Zweck der OR-Verknüpfung von Indizes in der Eliminierung doppelter RIDs liegt, besteht das Ziel der *logischen AND-Verknüpfung von Indizes* (Index ANDing) darin, gemeinsame RIDs zu finden. Die AND-Verknüpfung von Indizes kann auftreten, wenn Anwendungen mehrere Indizes für entsprechende Spalten innerhalb derselben Tabelle erstellen und eine Abfrage mit mehreren AND-Vergleichselementen für die Tabelle ausgeführt wird. Mehrere Indexsuchen für alle mit Indizes versehenen Spalten in einer solchen Abfrage ergeben Werte, mit denen in einem Hashverfahren Bitzuordnungen erstellt werden. Die zweite Bitzuordnung wird zur Prüfung der ersten Bitzuordnung verwendet, um die gesuchten Zeilen zu generieren, die zur Erstellung der endgültigen zurückzugebenden Datenmenge abgerufen werden.

Betrachten Sie zum Beispiel die beiden folgenden Indexdefinitionen:

```
INDEX IX4: SALARY   ASC
INDEX IX5: COMM     ASC
```

Die folgenden Vergleichselemente könnten mit Hilfe dieser beiden Indizes aufgelöst werden:

```
WHERE SALARY BETWEEN 20000 AND 30000
   AND COMM BETWEEN 1000 AND 3000
```

In diesem Beispiel generiert das Durchsuchen des Index `IX4` eine Bitzuordnung, die das Vergleichselement `SALARY BETWEEN 20000 AND 30000` erfüllt. Das Durchsuchen von `IX5` und Prüfen gegen die Bitzuordnung für `IX4` liefert eine Liste von RIDs, die beide Vergleichselemente erfüllen. Dies wird als „dynamische AND-Verknüpfung über Bitzuordnungen" bezeichnet. Diese Methode wird nur angewandt, sofern die Tabelle ausreichend Kardinalität hat und die Spalten genügend Werte in dem gesuchten Bereich oder genügend Duplizität haben, wenn Gleichheitsvergleichselemente verwendet werden.

Zur Umsetzung der Leistungsvorteile dynamischer Bitzuordnungen beim Durchsuchen mehrerer Indizes kann es notwendig sein, den Wert des Konfigurationsparameters für die Größe des Sortierspeichers (*sortheap*) der Datenbank und des Konfigurationsparameters für den Schwellenwert für Sortierspeicher (*sheapthres*) des Datenbankmanagers zu ändern.

Bei Verwendung dynamischer Bitzuordnungen in Zugriffsplänen wird zusätzlicher Sortierspeicher benötigt. Wenn der Wert von *sheapthres* relativ nahe am Wert von *sortheap* liegt (d. h. weniger als ein Faktor von 2- oder 3-mal pro gleichzeitiger Abfrage), steht dynamischen Bitzuordnungen mit Zugriff über mehrere Indizes wesentlich weniger Speicher zur Verfügung als das Optimierungsprogramm veranschlagt hat. Die Lösung besteht darin, den Wert von *sheapthres* relativ zu *sortheap* zu erhöhen.

Anmerkung: Das Optimierungsprogramm verwendet beim Zugriff auf eine einzelne Tabelle keine Kombination aus AND- und OR-Verknüpfungen von Indizes.

Indexzugriff in Bereichsclustertabellen

Im Gegensatz zu Standardtabellen benötigt eine Bereichsclustertabelle keinen physischen Index, der einer Zeile einen Schlüsselwert wie in einem traditionellen B-Baumstrukturindex zuordnet. Stattdessen greift er auf die sequenzielle Anordnung des Spaltenwertebereichs zurück und verwendet eine Zuordnungsfunktion, um die Position einer bestimmten Zeile in einer Tabelle zu generieren. Im einfachsten Beispiel einer solchen Zuordnung ist der erste Schlüsselwert im Bereich die erste Zeile in der Tabelle, der zweite Wert im Bereich die zweite Zeile in der Tabelle usw.

Das Optimierungsprogramm verwendet das Merkmal des Bereichsclustering der Tabelle, um Zugriffspläne auf der Basis eines perfekt angeordneten Index zu generieren, der lediglich den Aufwand zur Berechnung der Bereichsclusterfunktion erfordert. Das Clustering von Zeilen innerhalb der Tabelle ist gewährleistet, da Bereichsclustertabellen ihre Reihenfolge nach den ursprünglichen Schlüsselwerten beibehalten.

Zugehörige Konzepte:

- „Vor- und Nachteile von Indizes" auf Seite 288
- „Datenzugriffsmethoden" auf Seite 174
- „Datenzugriff über Indexsuchen" auf Seite 174
- „Indexzugriff und Clusterverhältnisse" auf Seite 180

Indexzugriff und Clusterverhältnisse

Bei der Auswahl des Zugriffsplans schätzt das Optimierungsprogramm die Anzahl der E/A-Operationen ab, die zum Einlesen der benötigten Seiten von der Platte in den Pufferpool erforderlich sind. Diese 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 E/A-Operationen anfallen.

Für Indexsuchen wird das Optimierungsprogramm durch Informationen aus den Systemkatalogtabellen (SYSCAT.INDEXES) bei der Abschätzung des Ein-/Ausgabeaufwands zum Lesen von Datenseiten in den Pufferpool unterstützt. Dabei werden Informationen aus den folgenden Spalten der Tabelle SYSCAT.INDEXES verwendet:

- Die Informationen der Spalte CLUSTERRATIO geben den Grad an, zu dem die Tabellendaten in Relation zu diesem Index in Clustern zusammengefasst sind (Clusterbildung). Je höher der Wert, desto besser sind die Zeilen in der Reihenfolge des Indexschlüssels geordnet. Wenn Tabellenzeilen nahe an der Reihenfolge des Indexschlüssels vorliegen, können Zeilen von einer Datenseite gelesen werden, während sich die Seite im Puffer befindet. Wenn der Wert dieser Spalte -1 ist, verwendet das Optimierungsprogramm die Informationen der Spalten PAGE_FETCH_PAIRS und CLUSTERFACTOR, wenn diese verfügbar sind.
- Die Spalte PAGE_FETCH_PAIRS enthält Paare von Zahlen, die zusammen mit CLUSTERFACTOR-Informationen jeweils ein Modell für die Anzahl der E/A-Operationen zum Lesen der Datenseiten in Pufferpools verschiedener Größen angeben. Für diese Spalten werden Daten nur erfasst, wenn Sie das Dienstprogramm RUNSTATS für den Index mit der Klausel DETAILED ausführen.

Wenn keine Statistiken zur Clusterbildung verfügbar sind, verwendet das Optimierungsprogramm Standardwerte, die von einem geringen Grad an Clusterbildung der Daten bezüglich des Index ausgehen.

Der Grad, zu dem die Daten in Bezug auf einen Index in Clustern zusammengefasst sind, kann bedeutende Auswirkungen auf die Leistung haben, so dass einer der Indizes für eine Tabelle auf einem Grad nahe an 100% Clusterbildung gehalten werden sollte.

Im Allgemeinen kann nur ein Index eine 100-prozentige Clusterbildung aufweisen. Eine Ausnahme bilden nur solche Fälle, in denen die Schlüssel eines anderen Index eine Obermenge der Schlüssel des Clusterindex sind oder in denen es eine De-Facto-Korrelation zwischen den Schlüsselspalten der beiden Indizes gibt.

Bei der Reorganisation einer Tabelle können Sie einen Index angeben, über den die Zeilen in Clustern angeordnet werden und versucht wird, diese Clusterbildung bei der Verarbeitung von Einfügungen beizubehalten. Da Aktualisierungen und Einfügungen die Clusterbildung in Bezug auf den Index verringern können, müssen Sie die Tabelle eventuell in regelmäßigen Abständen reorganisieren. Um die Häufigkeit der Reorganisation einer Tabelle zu verringern, die aufgrund von INSERT-, UPDATE- und DELETE-Anweisungen häufig geändert wird, können Sie beim Ändern einer Tabelle (ALTER TABLE) den Parameter PCTFREE verwenden. Dieser Parameter ermöglicht es, weitere Einfügungen so in die vorhandenen Daten einzugliedern, dass die Clusterbildung erhalten bleibt.

Zugehörige Konzepte:

- „Indexleistung - Tipps" auf Seite 293
- „Arten des Indexzugriffs" auf Seite 178

Terminologie für Vergleichselemente

Eine Benutzeranwendung fordert eine Menge von Zeilen aus der Datenbank mit Hilfe einer SQL-Anweisung an, in der die Ergebnismenge der gewünschten Zeilen mit Hilfe von Qualifikationsbedingungen angegeben werden. Diese Qualifikationsbedingungen stehen in der Regel in der WHERE-Klausel der Abfrage. Solche Qualifikationsbedingungen werden als *Vergleichselemente* bezeichnet. Vergleichselemente können in vier Kategorien zusammengefasst werden, die dadurch bestimmt sind, wie und wann das jeweilige Vergleichselement im Auswertungsprozess verwendet wird. Die Kategorien werden im Folgenden in der Reihenfolge von der höchsten bis zur niedrigsten Leistung geordnet aufgelistet:

1. Bereichsbegrenzende Vergleichselemente
2. Bei Indexsuchen als Suchargument verwendbare Vergleichselemente (Index SARGable)
3. Bei Datensuchen als Suchargument verwendbare Vergleichselemente (Data SARGable)
4. Restvergleichselemente

Anmerkung: Die Bezeichnung *SARGable* ist aus dem Begriff *s*earch *arg*ument abgeleitet.

Die folgende Tabelle enthält eine Übersicht über die Kategorien von Vergleichselementen. In den nachfolgenden Abschnitten werden die einzelnen Kategorien eingehender beschrieben.

Tabelle 27. Zusammenfassung der Merkmale der Vergleichselementkategorien

Merkmal	**Vergleichselementkategorie**			
	Bereichs-begrenzend	**Indexsuch-argument**	**Datensuch-argument**	**Restvergleichs-element**
Verringern der Index-Ein-/Ausgabe	Ja	Nein	Nein	Nein
Verringern der Datenseiteneinein-/ausgabe	Ja	Ja	Nein	Nein
Verringern der Anzahl intern übergebener Zeilen	Ja	Ja	Ja	Nein
Verringern der Anzahl der den Bedingungen entsprechenden Zeilen	Ja	Ja	Ja	Ja

Bereichsbegrenzende und bei Indexsuchen als Suchargumente verwendbare Vergleichselemente

Bereichsbegrenzende Vergleichselemente begrenzen den Bereich einer Indexsuche. Sie geben Start- und Stoppschlüsselwerte für die Indexsuche an. Bei Indexsuchen als Suchargument verwendbare Vergleichselemente können nicht den Bereich einer Suche begrenzen, aber sie können mit Hilfe des Index ausgewertet werden, da die im Vergleichselement verwendeten Spalten Teil des Indexschlüssels sind. Betrachten Sie zum Beispiel den folgenden Index:

```
INDEX IX1:  NAME    ASC,
            DEPT    ASC,
            MGR     DESC,
            SALARY  DESC,
            YEARS   ASC
```

Betrachten Sie dazu eine Abfrage, welche die folgende WHERE-Klausel enthält:

```
WHERE  NAME  = :hv1
 AND   DEPT  = :hv2
 AND   YEARS > :hv5
```

Die ersten beiden Vergleichselemente (NAME = :hv1, DEPT = :hv2) sind bereichsbegrenzende Vergleichselemente, während YEARS > :hv5 ein bei Indexsuchen als Suchargument verwendbares Vergleichselement ist.

Das Optimierungsprogramm verwendet die Indexdaten bei der Auswertung dieser Vergleichselemente, anstatt die Basistabelle zu lesen. Diese in Indizes als Suchargumente verwendbaren (*Index SARGable*) Vergleichselemente verringern die Anzahl der Zeilen, die aus der Tabelle gelesen werden müssen, sie beeinflussen jedoch nicht die Anzahl von Indexseiten, auf die zugegriffen wird.

Bei Datensuchen als Suchargument verwendbare Vergleichselemente

Vergleichselemente, die nicht vom Indexmanager ausgewertet werden können, sondern nur von den Datenverwaltungsservices, werden als bei Datensuchen verwendbare (*Data SARGable*) Vergleichselemente bezeichnet. Für solche Vergleichselemente ist in der Regel ein Zugriff auf einzelne Zeilen einer Tabelle erforderlich. Bei Bedarf rufen die Datenverwaltungsservices die zur Auswertung des Vergleichselements benötigten Spalten und andere Spalten ab, um die Spalten für die SELECT-Liste, die nicht aus dem Index abgerufen werden konnten, zur Verfügung zu stellen.

Betrachten Sie zum Beispiel einen einzelnen Index, der für die Tabelle PROJECT definiert ist:

```
INDEX IX0:  PROJNO ASC
```

Für die folgende Abfrage wird dann das Vergleichselement `DEPTNO = 'D11'` als bei Datensuchen als Suchargument verwendbar betrachtet.

```
SELECT PROJNO, PROJNAME, RESPEMP
  FROM PROJECT
 WHERE DEPTNO = 'D11'
 ORDER BY PROJNO
```

Restvergleichselemente

Restvergleichselemente erfordern mehr E/A-Aufwand als der Zugriff auf eine Tabelle. Sie können folgende Merkmale aufweisen:

- Sie verwenden korrelierte Unterabfragen.
- Sie verwenden quantifizierte Unterabfragen mit den Klauseln ANY, ALL, SOME oder IN.
- Sie Lesen LONG VARCHAR- oder LOB-Daten, die in einer von der Tabelle getrennten Datei gespeichert sind.

Solche Vergleichselemente werden von den Services für relationale Daten (Relational Data Services) ausgewertet.

Manchmal müssen Vergleichselemente, die nur auf den Index angewandt wurden, erneut angewandt werden, wenn auf die Datenseite zugegriffen wird. Zum Beispiel wenden Zugriffspläne mit OR-Verknüpfung oder AND-Verknüpfung von Indizes die Vergleichselemente immer ein weiteres Mal als Restvergleichselemente an, wenn auf die Datenseite zugegriffen wird.

Zugehörige Konzepte:

- „Der SQL-Compilerprozess” auf Seite 157

Verknüpfungsmethoden und -strategien

Dieser Abschnitt beschreibt, wie das Optimierungsprogramm Tabellen in der erforderlichen Weise verknüpft, um Ergebnisse an eine Abfrage zurückzuliefern, und erläutert die Methoden und Strategien, die vom Optimierungsprogramm angewendet werden.

Verknüpfungen

Eine *Verknüpfung* (engl. join) ist der Prozess der Kombination von Informationen aus mindestens zwei Tabellen auf der Grundlage eines gemeinsamen Geltungsbereichs der Informationen. Zeilen aus der einen Tabelle werden mit Zeilen aus einer anderen Tabelle gepaart, wenn die Informationen in den entsprechenden Zeilen die Verknüpfungsbedingung erfüllen.

Betrachten Sie zum Beispiel die beiden folgenden Tabellen:

Tabelle1		Tabelle2	
PROJ	**PROJ_ID**	**PROJ_ID**	**NAME**
A	1	1	Sam
B	2	3	Joe
C	3	4	Mary
D	4	1	Sue
		2	Mike

Zur Verknüpfung von Tabelle1 und Tabelle2 an den Stellen, an denen die ID-Spalten die gleichen Werte haben, verwenden Sie die folgende SQL-Anweisung:

```
SELECT PROJ, x.PROJ_ID, NAME
  FROM TABELLE1 x, TABELLE2 y
  WHERE x.PROJ_ID = y.PROJ_ID
```

Diese Anweisung liefert die folgende Menge von Ergebniszeilen:

PROJ	PROJ_ID	NAME
A	1	Sam
A	1	Sue
B	2	Mike
C	3	Joe
D	4	Mary

Abhängig davon, ob ein Verknüpfungsvergleichselement vorhanden ist und welche Verarbeitungsaufwände anhand der Tabellen- und Indexstatistiken ermittelt werden, wählt das Optimierungsprogramm eine der folgenden Verknüpfungsmethoden aus:

- Verknüpfung über Verschachtelungsschleife (Nested-Loop Join)
- Mischverknüpfung (Merge Join)
- Hashverknüpfung (Hash Join)

Bei der Verknüpfung zweier Tabellen wird die eine Tabelle als äußere Tabelle und die andere als innere Tabelle ausgewählt. Auf die äußere Tabelle wird zuerst zugegriffen, und sie wird nur einmal durchsucht. Ob die innere Tabelle mehrere Male durchsucht wird, hängt von der Art der Verknüpfung und den vorhandenen Indizes ab. Auch wenn in einer Abfrage mehr als zwei Tabellen verknüpft werden, ver-

knüpft das Optimierungsprogramm jeweils nur zwei Tabellen gleichzeitig. Falls erforderlich, werden temporäre Tabellen zum Speichern von Zwischenergebnissen erstellt.

Sie können explizite Verknüpfungsoperatoren wie INNER oder LEFT OUTER JOIN angeben, um festzulegen, wie Tabellen in der Verknüpfung verwendet werden. Bevor Sie jedoch eine Abfrage auf diese Art ändern, sollten Sie das Optimierungsprogramm ermitteln lassen, wie die Tabellen zu verknüpfen sind. Anschließend analysieren Sie die Abfrageleistung und entscheiden, ob Verknüpfungsoperatoren hinzugefügt werden sollten.

Zugehörige Konzepte:

- „Verknüpfungsmethoden" auf Seite 185
- „Verknüpfungsstrategien in partitionierten Datenbanken" auf Seite 193
- „Verknüpfungsmethoden in partitionierten Datenbanken" auf Seite 195
- „Verknüpfungsinformationen" auf Seite 654

Verknüpfungsmethoden

Das Optimierungsprogramm kann eine von drei Grundstrategien wählen, wenn Abfragen eine Verknüpfung von Tabellen erfordern.

- Verknüpfung über Verschachtelungsschleife (Nested-Loop Join)
- Mischverknüpfung (Merge Join)
- Hashverknüpfung (Hash Join)

Diese Methoden werden in den folgenden Abschnitten beschrieben.

Verknüpfung über Verschachtelungsschleife

Eine Verknüpfung über Verschachtelungsschleife (Nested-Loop Join) wird auf eine der beiden folgenden Arten ausgeführt:

- Durchsuchen der inneren Tabelle für jede Zeile der äußeren Tabelle, auf die zugegriffen wird

  Betrachten Sie zum Beispiel die Spalte A in den Tabellen T1 und T2 mit folgenden Werten:

Äußere Tabelle T1: Spalte A	Innere Tabelle T2: Spalte A
2	3
3	2
3	2
	3
	1

  Zur Durchführung einer Verknüpfung über Verschachtelungsschleife geht der Datenbankmanager in folgenden Schritten vor:

  1. Lesen der ersten Zeile aus T1. Der Wert für A ist „2".
  2. Durchsuchen von T2, bis ein übereinstimmender Wert („2") gefunden wird, und anschließendes Verknüpfen der beiden Zeilen.
  3. Durchsuchen von T2, bis der nächste übereinstimmende Wert („2") gefunden wird, und anschließendes Verknüpfen der beiden Zeilen.
  4. Durchsuchen von T2 bis zum Ende der Tabelle.
  5. Zurückkehren zu T1 und Lesen der nächsten Zeile („3").

6. Durchsuchen von T2 von der ersten Zeile an, bis ein übereinstimmender Wert („3“) gefunden wird, und anschließendes Verknüpfen der beiden Zeilen.
7. Durchsuchen von T2, bis der nächste übereinstimmende Wert („3“) gefunden wird, und anschließendes Verknüpfen der beiden Zeilen.
8. Durchsuchen von T2 bis zum Ende der Tabelle.
9. Zurückkehren zu T1 und Lesen der nächsten Zeile („3“).
10. Durchsuchen von T2 wie vorher und Verknüpfen aller übereinstimmenden Zeilen („3“).

- Durchführen einer Indexsuche für die innere Tabelle für jede Zeile der äußeren Tabelle, auf die zugegriffen wird

  Diese Methode kann für die angegebenen Vergleichselemente verwendet werden, wenn es ein Vergleichselement der folgenden Form gibt:

  ```
  ausdr(äußere_tabelle.spalte)  relop  innere_tabelle.spalte
  ```

  Dabei ist `relop` ein relativer Operator (z. B. =, >, >=, < oder <=) und `ausdr` ist ein gültiger Ausdruck für die äußere Tabelle. Betrachten Sie die folgenden Beispiele:

  ```
  OUTER.C1 + OUTER.C2 <= INNER.C1
  OUTER.C4 < INNER.C3
  ```

  Diese Methode könnte die Anzahl der Zeilen, auf die in der inneren Tabelle für jeden Zugriff auf die äußere Tabelle zugegriffen wird, wesentlich verringern, obwohl dies von einer Reihe von Faktoren abhängig ist, zu denen auch die Selektivität des Verknüpfungsvergleichselements zählt.

Bei der Beurteilung einer Verknüpfung über Verschachtelungsschleife entscheidet das Optimierungsprogramm auch, ob die äußere Tabelle sortiert wird oder nicht, bevor die Verknüpfung durchgeführt wird. Durch Sortieren der äußeren Tabelle nach den Werten der Verknüpfungsspalten kann die Anzahl der Leseoperationen zum Zugriff auf Seiten auf der Platte für die innere Tabelle verringert werden, da es wahrscheinlicher wird, dass sich die Seiten bereits im Pufferpool befinden. Wenn die Verknüpfung einen Index mit einem hohen Grad der Clusterbildung verwendet, um auf die innere Tabelle zuzugreifen, und wenn die äußere Tabelle sortiert wurde, kann die Anzahl von Indexseiten, auf die zugegriffen wird, minimiert werden.

Wenn darüber hinaus das Optimierungsprogramm erwartet, dass durch die Verknüpfung eine spätere Sortierung aufwendiger wird, kann es entscheiden, die Sortierung vor der Verknüpfung durchzuführen. Eine spätere Sortierung könnte erforderlich sein, um die Klauseln GROUP BY, DISTINCT, ORDER BY oder eine Mischverknüpfung zu unterstützen.

Mischverknüpfung

Eine Mischverknüpfung (engl. Merge Join, Merge Scan Join oder Sort Merge Join) erfordert ein Vergleichselement der Form tabelle1.spalte = tabelle2.spalte. Ein solches Vergleichselement wird als *Gleichheitsverknüpfungselement* bezeichnet. Eine Mischverknüpfung erfordert entweder über einen Indexzugriff oder durch eine Sortierung geordnete Eingaben für die Verknüpfungsspalten. Eine Mischverknüpfung kann nicht verwendet werden, wenn die Verknüpfungsspalte eine LONG-Feldspalte oder eine LOB-Spalte ist.

Bei einer Mischverknüpfung werden die Tabellen, die verknüpft werden, gleichzeitig durchsucht. Die äußere Tabelle der Mischverknüpfung wird nur einmal durch-

sucht. Die innere Tabelle wird auch nur einmal durchsucht, sofern in der äußeren Tabelle keine sich wiederholenden Werte auftreten. Wenn wiederholte Werte auftreten, wird eine Gruppe von Zeilen der inneren Tabelle eventuell noch einmal durchsucht. Betrachten Sie zum Beispiel die Spalte A in den Tabellen T1 und T2 mit folgenden Werten:

Äußere Tabelle T1: Spalte A	Innere Tabelle T2: Spalte A
2	1
3	2
3	2
	3
	3

Zur Durchführung einer Mischverknüpfung geht der Datenbankmanager in folgenden Schritten vor:

1. Lesen der ersten Zeile aus T1. Der Wert für A ist „2".
2. Durchsuchen von T2, bis ein übereinstimmender Wert gefunden wird, und anschließendes Verknüpfen der beiden Zeilen.
3. Fortsetzen des Durchsuchens von T2, solange die Spalten übereinstimmen, und dabei Verknüpfen der Zeilen.
4. Wenn der Wert „3" in T2 gelesen wird, Zurückkehren zu T1 und Lesen der nächsten Zeile.
5. Der nächste Wert in T1 ist „3", der mit T2 übereinstimmt, also Verknüpfen der Spalten.
6. Fortsetzen des Durchsuchens von T2, solange die Spalten übereinstimmen, und dabei Verknüpfen der Zeilen.
7. Erreichen des Endes von T2.
8. Zurückkehren zu T1, um die nächste Zeile zu lesen. Beachten Sie, dass der nächste Wert in T1 derselbe ist wie der vorige Wert aus T1, so dass T2 noch einmal, angefangen vom ersten Wert „3" in T2, durchsucht wird. Der Datenbankmanager speichert diese Position.

Hashverknüpfung

Eine Hashverknüpfung erfordert ein oder mehrere Vergleichselemente der Form tabelle1.spalteX = tabelle2.spalteY, wobei die Spaltentypen übereinstimmen müssen. Spalten des Typs CHAR müssen die gleiche Länge aufweisen. Bei Spalten des Typs DECIMAL muss die Genauigkeit und die Anzahl der Kommastellen übereinstimmen. Der Spaltentyp kann keine LONG-Feldspalte oder LOB-Spalte sein.

Zunächst wird die als INNER designierte Tabelle (innere Tabelle) durchsucht, wobei die Zeilen in Speicherpuffer kopiert werden, die dem Sortierzwischenspeicher entnommen werden, der wiederum durch den Datenbankkonfigurationsparameter *sortheap* definiert ist. Die Speicherpuffer werden auf der Grundlage eines Hashwerts, der auf den Spalten der Verknüpfungsvergleichselemente berechnet wird, in Partitionen unterteilt. Wenn die Größe der inneren Tabelle (INNER) die Größe des Sortierspeichers überschreitet, werden Puffer aus ausgewählten Partitionen in temporäre Tabellen geschrieben.

Wenn die innere Tabelle verarbeitet wurde, wird die zweite, als OUTER designierte Tabelle (äußere Tabelle) durchsucht und ihre Zeilen werden mit den Zeilen aus der inneren Tabelle (INNER) abgeglichen, indem zuerst der Hashwert, der für die Spalten der Verknüpfungsvergleichselemente errechnet wurde, verglichen wird. Wenn

der Hashwert für die OUTER-Zeilenspalte mit dem Hashwert der INNER-Zeilenspalte übereinstimmt, werden die tatsächlichen Werte der Verknüpfungselementspalten verglichen.

Zeilen der äußeren Tabelle, die Partitionen entsprechen, die nicht in eine temporäre Tabelle geschrieben wurden, werden sofort mit den Zeilen der inneren Tabelle im Speicher abgeglichen. Wenn die entsprechende Partition der inneren Tabelle in eine temporäre Tabelle geschrieben wurde, wird die OUTER-Zeile ebenfalls in eine temporäre Tabelle geschrieben. Schließlich werden übereinstimmende Paare von Partitionen aus den temporären Tabellen gelesen, die Hashwerte ihrer Zeilen abgeglichen und die Verknüpfungsvergleichselemente geprüft.

Zur Erzielung der vollen Leistungsvorteile der Hashverknüpfung müssen Sie möglicherweise den Wert des Konfigurationsparameters *sortheap* der Datenbank und des Konfigurationsparameters *sheapthres* des Datenbankmanagers ändern.

Die Leistung der Hashverknüpfung ist am besten, wenn Sie Hashschleifen und Überläufe auf den Plattenspeicher vermeiden können. Zur Optimierung der Leistung von Hashverknüpfungen schätzen Sie die maximale Speichergröße ab, die für *sheapthres* verfügbar ist, und optimieren dann den Parameter **sortheap**. Erhöhen Sie den Wert dieses Parameters, bis Sie möglichst viele Hashschleifen und Überläufe auf den Plattenspeicher vermeiden, jedoch nicht den durch den Parameter *sheapthres* definierten Grenzwert erreichen.

Die Erhöhung des Werts für *sortheap* sollte außerdem die Leistung von Abfragen verbessern, die mehrere Sortierungen erfordern.

Zugehörige Konzepte:

- „Verknüpfungen" auf Seite 184
- „Verknüpfungsstrategien in partitionierten Datenbanken" auf Seite 193
- „Verknüpfungsmethoden in partitionierten Datenbanken" auf Seite 195
- „Verknüpfungsinformationen" auf Seite 654

Zugehörige Referenzen:

- „sortheap - Zwischenspeichergröße für Sortierlisten" auf Seite 418
- „sheapthres - Schwellenwert für Sortierspeicher" auf Seite 416

Strategien zur Auswahl optimaler Verknüpfungen

Das Optimierungsprogramm nutzt verschiedene Methoden zur Auswahl einer optimalen Verknüpfungsstrategie für eine Abfrage. Zu diesen Methoden zählen die folgenden Suchstrategien, die durch die Optimierungsklasse der Abfrage bestimmt werden:

- Schnelle Verknüpfungsaufzählung (Greedy Join Enumeration)
  - Effizient im Hinblick auf Speicherbedarf und Zeit.
  - Aufzählung in einer Richtung, d. h., wenn eine Verknüpfungsmethode für zwei Tabellen ausgewählt ist, wird sie im Laufe der weiteren Optimierung nicht mehr geändert.
  - Eventuell wird nicht der optimale Zugriffsplan gewählt, wenn viele Tabellen verknüpft werden. Wenn eine Abfrage nur zwei oder drei Tabellen verknüpft, ist der Zugriffsplan, der durch die schnelle Verknüpfungsaufzählung ausgewählt wird, derselbe wie der Zugriffsplan, der durch die dynamisch programmierte Verknüpfungsaufzählung (Dynamic Programming Join Enumeration)

ausgewählt wird. Dies gilt insbesondere dann, wenn die Abfrage viele entweder explizit angegebene oder implizit durch die Anwendung der Transitivität generierte Verknüpfungsvergleichselemente für dieselbe Spalte hat.

- Dynamisch programmierte Verknüpfungsaufzählung (Dynamic Programming Join Enumeration)
  - Der Bedarf an Speicherplatz und Zeit wächst exponenziell mit zunehmender Anzahl der verknüpften Tabellen.
  - Effiziente und erschöpfende Suche nach dem besten Zugriffsplan.
  - Ähnlich der Strategie, die von DB2® für OS/390 oder z/OS verwendet wird.

Der Algorithmus für die Verknüpfungsaufzählung spielt die entscheidende Rolle bei der Bestimmung der Anzahl von Zugriffsplankombinationen, die das Optimierungsprogramm prüft.

Sternschemaverknüpfungen

Die Tabellen, auf die in einer Abfrage verwiesen wird, sind fast immer durch Verknüpfungsvergleichselemente miteinander verbunden. Wenn zwei Tabellen ohne Verknüpfungsvergleichselement verknüpft werden, wird das kartesische Produkt der beiden Tabellen gebildet. Bei einem kartesischen Produkt wird jede ausgewählte Zeile der ersten Tabelle mit jeder ausgewählten Zeile der zweiten Tabelle verknüpft. Das Ergebnis ist eine Tabelle, die aus dem Kreuzprodukt in der Größe der beiden Tabellen besteht und in der Regel sehr groß ist. Da ein solcher Plan wahrscheinlich keine gute Leistung zulässt, vermeidet das Optimierungsprogramm sogar die Aufwandsabschätzung für einen Plan dieser Art.

Die einzigen Ausnahmen bilden die Fälle, in denen die Optimierungsklasse auf 9 gesetzt wird oder der Sonderfall eines Sternschemas vorliegt. Ein *Sternschema* (engl. star schema) enthält eine zentrale Tabelle, die als Fakttabelle bezeichnet wird, und andere Tabellen, die als Dimensionstabellen bezeichnet werden. Die Dimensionstabellen haben, ungeachtet der Abfrage, jeweils nur eine einzige Verknüpfung, über die sie mit der Fakttabelle verbunden sind. Jede Dimensionstabelle enthält zusätzliche Werte, die die Informationen über eine bestimmte Spalte in der Fakttabelle erweitern. Eine typische Abfrage besteht aus mehreren lokalen Vergleichselementen, die auf Werte in den Dimensionstabellen verweisen, und enthält Verknüpfungsvergleichselemente, die die Dimensionstabellen mit der Fakttabelle verbinden. Für solche Abfragen kann es vorteilhaft sein, das kartesische Produkt mehrerer kleiner Dimensionstabellen zu berechnen und erst anschließend auf die umfangreiche Fakttabelle zuzugreifen. Diese Technik ist dann von Vorteil, wenn mehrere Verknüpfungsvergleichselemente einem mehrspaltigen Index entsprechen.

DB2 kann Abfragen erkennen, die für Datenbanken durchgeführt werden, die mit Sternschemata aufgebaut sind und mindestens zwei Dimensionstabellen haben, und kann den Suchbereich vergrößern, um mögliche Zugriffspläne mit kartesischen Produkten von Dimensionstabellen zu berücksichtigen. Wenn der Plan mit den kartesischen Produkten den niedrigsten geschätzten Aufwand verursacht, wird er vom Optimierungsprogramm ausgewählt.

Die oben behandelte Strategie der Sternschemaverknüpfung basiert auf der Annahme, dass Primärschlüsselindizes in der Verknüpfung verwendet werden. Eine andere Situation liegt vor, wenn Fremdschlüsselindizes verwendet werden. Wenn die Fremdschlüsselspalten in der Fakttabelle einspaltige Indizes sind und es eine relativ hohe Selektivität über alle Dimensionstabellen hinweg gibt, könnte die folgende Methode der Sternverknüpfung zur Anwendung kommen:

1. Verarbeiten jeder Dimensionstabelle wie folgt:

- Durchführen einer einfachen Gleichheitsverknüpfung zwischen der Dimensionstabelle und dem Fremdschlüsselindex für die Fakttabelle
- Dynamisches Erstellen einer Bitzuordnung durch ein Hashverfahren für die Werte der Zeilen-IDs (RID)

2. Anwenden von AND-Vergleichselementen auf die vorige Bitzuordnung für jede Bitzuordnung
3. Feststellen der verbliebenen RIDs, nachdem die letzte Bitzuordnung verarbeitet wurde
4. Optionales Sortieren dieser RIDs
5. Abrufen einer Zeile aus der Basistabelle
6. Wiederverknüpfen der Fakttabelle mit jeder der zugehörigen Dimensionstabellen, dabei Zugreifen auf die Spalten in Dimensionstabellen, die für die SELECT-Klausel benötigt werden
7. Erneutes Anwenden der Restvergleichselemente

Diese Methode erfordert keine mehrspaltigen Indizes. Explizite referenzielle Integritätsbedingungen zwischen Fakttabelle und Dimensionstabellen sind nicht erforderlich, obwohl die Beziehung zwischen Fakttabelle und Dimensionstabellen doch auf diese Weise realisiert werden sollte.

Die Bitzuordnungen, die von Sternverknüpfungstechniken dynamisch erstellt und verwendet werden, erfordern Sortierspeicher, dessen Größe durch den Datenbankkonfigurationsparameter *sortheap* definiert wird.

Zusammengesetzte Tabellen

Wenn das Ergebnis der Verknüpfung zweier Tabellen eine neue Tabelle ist, die in diesem Fall als *zusammengesetzte* Tabelle bezeichnet wird, wird diese Tabelle in der Regel zur äußeren Tabelle einer weiteren Verknüpfung mit einer anderen inneren Tabelle. Eine solche Verknüpfung wird als „Verknüpfung mit zusammengesetzter äußerer Tabelle" bezeichnet. In einigen Fällen, besonders bei Verwendung der schnellen Verknüpfungsaufzählung (Greedy Join Enumeration), ist es sinnvoll, das Ergebnis der Verknüpfung zweier Tabellen zur inneren Tabelle einer späteren Verknüpfung zu machen. Wenn die innere Tabelle einer Verknüpfung aus dem Ergebnis der Verknüpfung zweier oder mehrerer Tabellen besteht, wird ein solcher Plan als „Verknüpfung mit zusammengesetzter innerer Tabelle" bezeichnet. Betrachten Sie zum Beispiel die folgende Abfrage:

```
SELECT COUNT(*)
FROM T1, T2, T3, T4
WHERE T1.A = T2.A AND
      T3.A = T4.A AND
      T2.Z = T3.Z
```

Hier könnte es von Vorteil sein, die Tabellen T1 und T2 zu verknüpfen (T1xT2), anschließend die Tabellen T3 und T4 zu verknüpfen (T3xT4) und schließlich das Ergebnis der ersten Verknüpfung als äußere Tabelle und das Ergebnis der zweiten Verknüpfung als innere Tabelle auszuwählen. Im daraus resultierenden Plan ((T1xT2) x (T3xT4)) ist das Ergebnis der Verknüpfung (T3xT4) eine zusammengesetzte innere Tabelle. Abhängig von der Abfrageoptimierungsklasse belegt das Optimierungsprogramm die maximale Anzahl von Tabellen, die als innere Tabelle einer Verknüpfung verwendet werden können, mit unterschiedlichen Einschränkungen. Verknüpfungen mit zusammengesetzten inneren Tabellen sind bei den Optimierungsklassen 5, 7 und 9 zulässig.

Zugehörige Konzepte:
- „Verknüpfungen" auf Seite 184
- „Verknüpfungsmethoden" auf Seite 185
- „Verknüpfungsstrategien in partitionierten Datenbanken" auf Seite 193
- „Verknüpfungsmethoden in partitionierten Datenbanken" auf Seite 195

Replizierte gespeicherte Abfragetabellen in partitionierten Datenbanken

Replizierte gespeicherte Abfragetabellen verbessern die Leistung häufig ausgeführter Verknüpfungen in einer partitionierten Datenbankumgebung, indem sie der Datenbank die Möglichkeit geben, vorberechnete Werte der Tabellendaten zu pflegen. Betrachten Sie ein Beispiel einer Abfrage und einer replizierten gespeicherten Abfragetabelle. Dazu gelten folgende Annahmen:
- Die Tabelle SALES (Verkauf) befindet sich im Tabellenbereich REGIONTABLESPACE, der in mehrere Partitionen unterteilt ist, und ist über die Spalte REGION partitioniert.
- Die Tabellen EMPLOYEE (Mitarbeiter) und DEPARTMENT (Abteilung) sind in einer Datenbankpartitionsgruppe mit Einzelpartition.

Erstellen Sie eine replizierte gespeicherte Abfragetabelle auf der Basis der Informationen in der Tabelle EMPLOYEE.

```
CREATE TABLE R_EMPLOYEE
  AS (
      SELECT EMPNO, FIRSTNME, MIDINIT, LASTNAME, WORKDEPT
      FROM   EMPLOYEE
     )
DATA INITIALLY DEFERRED REFRESH IMMEDIATE
IN REGIONTABLESPACE
REPLICATED;
```

Führen Sie zur Aktualisierung der replizierten gespeicherten Abfragetabelle die folgende Anweisung aus:

```
REFRESH TABLE R_EMPLOYEE;
```

Anmerkung: Nach der Verwendung der Anweisung REFRESH sollten Sie RUNSTATS für die replizierte Tabelle in gleicher Weise wie für die anderen Tabellen ausgeführen.

Im folgenden Beispiel werden der Verkauf nach Mitarbeiter, die Summe für die Abteilung und die Gesamtsumme berechnet:

```
SELECT d.mgrno, e.empno, SUM(s.sales)
FROM   department AS d, employee AS e, sales AS s
WHERE  s.sales_person = e.lastname
   AND e.workdept = d.deptno
GROUP BY ROLLUP(d.mgrno, e.empno)
ORDER BY d.mgrno, e.empno;
```

Anstatt der Tabelle EMPLOYEE, die nur in einer Datenbankpartition ist, verwendet der Datenbankmanager die Tabelle R_EMPLOYEE, die in jede der Datenbankpartitionen repliziert wird, in der die Tabelle SALES gespeichert ist. Der Leistungsvorteil ergibt sich daraus, dass die Mitarbeiterinformationen zur Berechnung der Verknüpfung nicht zu jeder Datenbankpartition über das Netzwerk gesendet werden müssen.

Replizierte gespeicherte Abfragetabellen in zusammengefassten Verknüpfungen

Replizierte gespeicherte Abfragetabellen können auch bei der Kollokation von Verknüpfungen helfen. Wenn beispielsweise ein Sternschema eine große Fakttabelle enthält, die sich über zwanzig Knoten erstreckt, sind die Verknüpfungen zwischen der Fakttabelle und den Dimensionstabellen am effizientesten, wenn diese Tabellen durch Kollokation zusammenfasst werden. Wenn sich alle Tabellen in derselben Datenbankpartitionsgruppe befinden, ist höchstens eine Dimensionstabelle korrekt für eine zusammengefasste Verknüpfung (d. h. Verknüpfung von durch Kollokation zusammengefasster Tabellen) partitioniert. Die anderen Dimensionstabellen können nicht in einer zusammengefassten Verknüpfung verwendet werden, weil die Verknüpfungsspalten der Fakttabelle nicht dem Partitionierungsschlüssel der Fakttabelle entsprechen.

Betrachten Sie zum Beispiel eine Tabelle mit dem Namen FACT (C1, C2, C3, ...), die über Spalte C1 partitioniert ist, und eine Tabelle mit dem Namen DIM1 (C1, dim1a, dim1b, ...), die über C1 partitioniert ist, und eine Tabelle DIM2 (C2, dim2a, dim2b, ...), die über C2 partitioniert ist, usw.

In diesem Fall können Sie erkennen, dass die Verknüpfung zwischen FACT und DIM1 perfekt ist, da das Vergleichselement DIM1.C1 = FACT.C1 durch Kollokation zusammengefasst vorliegt. Die beiden Tabellen werden über die Spalte C1 partitioniert.

Die Verknüpfung mit DIM2 mit dem Vergleichselement WHERE DIM2.C2 = FACT.C2 kann jedoch nicht durch Kollokation zusammengefasst werden, da FACT über die Spalte C1 partitioniert ist und nicht über die Spalte C2. In diesem Fall kann es sinnvoll sein, die Tabelle DIM2 in der Datenbankpartitionsgruppe der Fakttabelle zu replizieren, so dass die Verknüpfung lokal in jeder Partition erfolgen kann.

Anmerkung: Diese Erörterung replizierter gespeicherter Abfragetabellen bezieht sich auf datenbankinterne Replikation. Datenbankübergreifende Replikation erfordert Subskriptionen, Steuertabellen und Daten in verschiedenen Datenbanken und auf verschiedenen Betriebssystemen.

Wenn Sie eine replizierte gespeicherte Abfragetabelle erstellen, kann die Quellentabelle eine Tabelle auf einem einzelnen Knoten oder eine Tabelle auf mehreren Knoten in einer Datenbankpartitionsgruppe sein. In den meisten Fällen ist die replizierte Tabelle klein und kann in einer Datenbankpartitionsgruppe mit einem Knoten untergebracht werden. Sie können die zu replizierende Datenmenge einschränken, indem Sie nur einen Teil der Spalten der Tabelle angeben, die Zeilenzahl mit Hilfe der verwendeten Vergleichselemente begrenzen oder beide Methoden anwenden. Für das Funktionieren replizierter gespeicherter Abfragetabellen ist die Option zur Datenerfassung nicht erforderlich.

Eine replizierte gespeicherte Abfragetabelle kann auch in einer Datenbankpartitionsgruppe mit mehreren Knoten erstellt werden, so dass Kopien der Quellentabelle in allen Partitionen erstellt werden. Verknüpfungen zwischen einer großen Fakttabelle und den Dimensionstabellen finden in dieser Art von Umgebung mit größerer Wahrscheinlichkeit lokal statt, als wenn die Quellentabelle an alle Partitionen per Broadcast übertragen werden muss.

Indizes für replizierte Tabellen werden nicht automatisch erstellt. Sie können Indizes erstellen, die sich von denen für die Quellentabelle unterscheiden. Zur Vermeidung von Verletzungen von Integritätsbedingungen, die für die Quellentabellen

nicht vorhanden sind, können Sie jedoch keine eindeutigen Indizes erstellen oder Integritätsbedingungen für die replizierten Tabellen definieren. Integritätsbedingungen sind auch dann nicht zulässig, wenn die gleichen Integritätsbedingungen für die Quellentabelle gelten.

Auf replizierte Tabellen kann in einer Abfrage direkt verwiesen werden, jedoch können Sie nicht das Vergleichselement NODENUMBER() mit einer replizierten Tabelle verwenden, um die Tabellendaten in einer bestimmten Partition abzurufen.

Verwenden Sie die EXPLAIN-Einrichtung, um festzustellen, ob eine replizierte gespeicherte Abfragetabelle von dem Zugriffsplan für eine Abfrage genutzt wurde. Ob der vom Optimierungsprogramm ausgewählte Zugriffsplan die replizierte gespeicherte Abfragetabelle verwendet, hängt von den Informationen ab, die verknüpft werden müssen. Die replizierte Übersichtstabelle könnte zum Beispiel dann nicht verwendet werden, wenn das Optimierungsprogramm feststellt, dass es weniger aufwendig wäre, die ursprüngliche Quellentabelle an die anderen Partitionen der Datenbankpartitionsgruppe per Broadcast rundzusenden.

Zugehörige Konzepte:

• „Verknüpfungen" auf Seite 184

Verknüpfungsstrategien in partitionierten Datenbanken

In mancher Hinsicht unterscheiden sich die Verknüpfungsstrategien in einer partitionierten Datenbank von denen in einer nicht partitionierten Datenbank. Zur Verbesserung der Leistung können zusätzliche Techniken auf die Standardverknüpfungsmethoden angewandt werden.

Ein Gesichtspunkt bei Tabellen, die häufig an Verknüpfungen in einer partitionierten Datenbank beteiligt sind, ist die Tabellenkollokation, d. h. die physische Zusammenfassung von Tabellen. Die Tabellenkollokation stellt eine Methode in einer partitionierten Datenbank dar, mit der Daten aus einer Tabelle mit Hilfe eines gleichen Partitionierungsschlüssels zusammen mit den Daten aus einer anderen Tabelle in derselben Partition angeordnet werden. Wenn die Daten einmal in dieser Weise zusammengefasst sind, können zu verknüpfende Daten an einer Abfrage beteiligt sein, ohne im Rahmen der Aktivitäten für die Abfrage von einer Datenbankpartition in eine andere übertragen werden zu müssen. Nur die Ergebnismenge einer Verknüpfung wird an den Koordinatorknoten übergeben.

Tabellenwarteschlangen

Die Beschreibungen von Verknüpfungstechniken in einer partitionierten Datenbank greifen auf folgende Terminologie zurück:

- Tabellenwarteschlange

  Ein Mechanismus zur Übertragung von Zeilen zwischen Datenbankpartitionen oder auch zwischen Prozessoren in einer Datenbank mit einer Einzelpartition.
- Gezielt übertragene Tabellenwarteschlange

  Eine Tabellenwarteschlange, in der Zeilen durch ein Hashverfahren gezielt an eine der empfangenden Datenbankpartitionen geleitet werden.
- Broadcast-Tabellenwarteschlange

  Eine Tabellenwarteschlange, in der Zeilen ohne Hashverfahren an alle empfangenden Datenbankpartitionen gesendet werden.

Eine Tabellenwarteschlange wird zu folgenden Zwecken verwendet:

- Übertragen von Tabellendaten von einer Datenbankpartition zu einer anderen bei Verwendung partitionsübergreifender Parallelität
- Übertragen von Tabellendaten innerhalb einer Datenbankpartition bei Verwendung partitionsinterner Parallelität
- Übertragen von Tabellendaten innerhalb einer Datenbankpartition bei Verwendung einer einzigen Datenbankpartition

Jede Tabellenwarteschlange überträgt die Daten in eine einzige Richtung. Der Compiler entscheidet, wo Tabellenwarteschlangen erforderlich sind, und nimmt sie in den Plan auf. Bei der Ausführung des Plans werden die Tabellenwarteschlangen durch die Verbindungen zwischen den Datenbankpartitionen initiiert. Die Tabellenwarteschlangen werden am Ende der Prozesse wieder geschlossen.

Es gibt verschiedene Arten von Tabellenwarteschlangen:

- *Asynchrone Tabellenwarteschlangen.* Diese Tabellenwarteschlangen werden als asynchron bezeichnet, weil sie Zeilen vor einer FETCH-Anweisung, die durch eine Anwendung abgesetzt wird, vorablesen. Wenn eine FETCH-Anweisung abgesetzt wird, wird die Zeile aus der Tabellenwarteschlange abgerufen.

  Asynchrone Tabellenwarteschlangen werden verwendet, wenn Sie die Klausel FOR FETCH ONLY in der SELECT-Anweisung verwenden. Wenn Sie Zeilen nur mit FETCH abrufen, ist eine asynchrone Tabellenwarteschlange schneller.
- *Synchrone Tabellenwarteschlangen.* Diese Tabellenwarteschlangen werden als synchron bezeichnet, weil sie für jede FETCH-Anweisung, die durch eine Anwendung abgesetzt wird, eine Zeile lesen. In jeder Datenbankpartition wird der Cursor in die nächste Zeile gesetzt, die aus der jeweiligen Datenbankpartition zu lesen ist.

  Synchrone Tabellenwarteschlangen werden verwendet, wenn Sie die Klausel FOR FETCH ONLY in der SELECT-Anweisung nicht angeben. In einer Umgebung mit partitionierten Datenbanken verwendet der Datenbankmanager synchrone Tabellenwarteschlangen, wenn Zeilen aktualisiert werden.
- *Tabellenwarteschlangen für Mischverknüpfung.*

  Bei dieser Art von Tabellenwarteschlange wird die Reihenfolge gewahrt.
- *Reguläre Tabellenwarteschlangen.*

  Diese Tabellenwarteschlangen sind nicht für Mischverknüpfungen geeignet. Bei ihnen bleibt die Reihenfolge nicht erhalten.
- *Empfangende Tabellenwarteschlangen.*

  Diese Tabellenwarteschlangen werden mit korrelierten Unterabfragen verwendet. Die Korrelationswerte werden an die Unterabfrage übergeben, und die Ergebnisse mit Hilfe dieser Art von Tabellenwarteschlange an den übergeordneten Abfrageblock zurückgeliefert.

Zugehörige Konzepte:

- „Verknüpfungen" auf Seite 184
- „Verknüpfungsmethoden" auf Seite 185
- „Verknüpfungsmethoden in partitionierten Datenbanken" auf Seite 195

Verknüpfungsmethoden in partitionierten Datenbanken

Die folgenden Abbildungen veranschaulichen Verknüpfungsmethoden in einer partitionierten Datenbank.

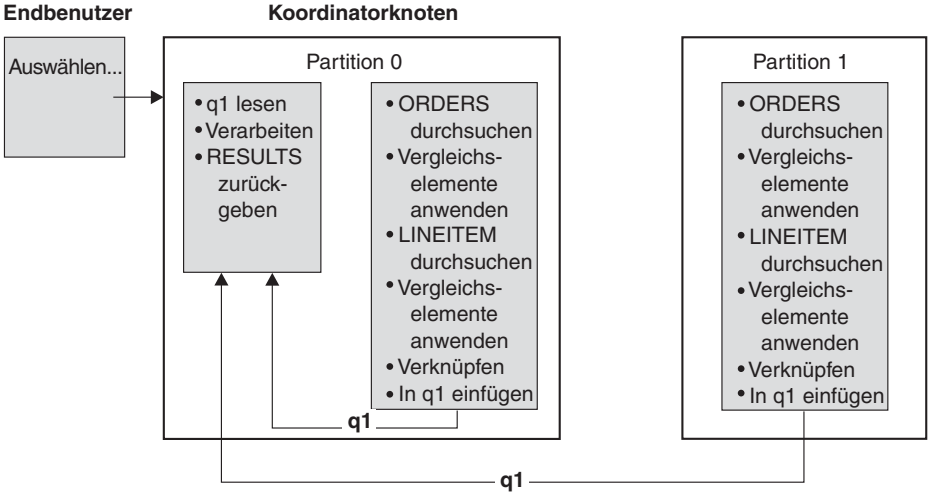
Anmerkung: In den Abbildungen beziehen sich q1, q2 und q3 (für „Queue") auf die Tabellenwarteschlangen in den Beispielen. Die Tabellen, auf die zugegriffen wird, sind für den Zweck dieser Beispielszenarios auf zwei Datenbankpartitionen verteilt. Die Pfeile zeigen die Richtung an, in der die Tabellenwarteschlangen übertragen bzw. gesendet werden. Der Koordinatorknoten ist Partition 0.

Zusammengefasste Verknüpfungen

Eine zusammengefasste Verknüpfung findet lokal in der Partition statt, in der sich die Daten befinden. Die Partition sendet die Daten an die anderen Partitionen, wenn die Verknüpfung abgeschlossen ist. Damit das Optimierungsprogramm eine zusammengefasste Verknüpfung in Erwägung ziehen kann, müssen die zu verknüpfenden Tabellen durch Kollokation zusammengefasst sein, und alle Paare mit dem entsprechenden Partitionierungsschlüssel müssen in den Gleichheitsverknüpfungselementen vertreten sein.

Die folgende Abbildung zeigt ein Beispiel.

Anmerkung: Replizierte gespeicherte Abfragetabellen erhöhen die Wahrscheinlichkeit zusammengefasster Verknüpfungen.

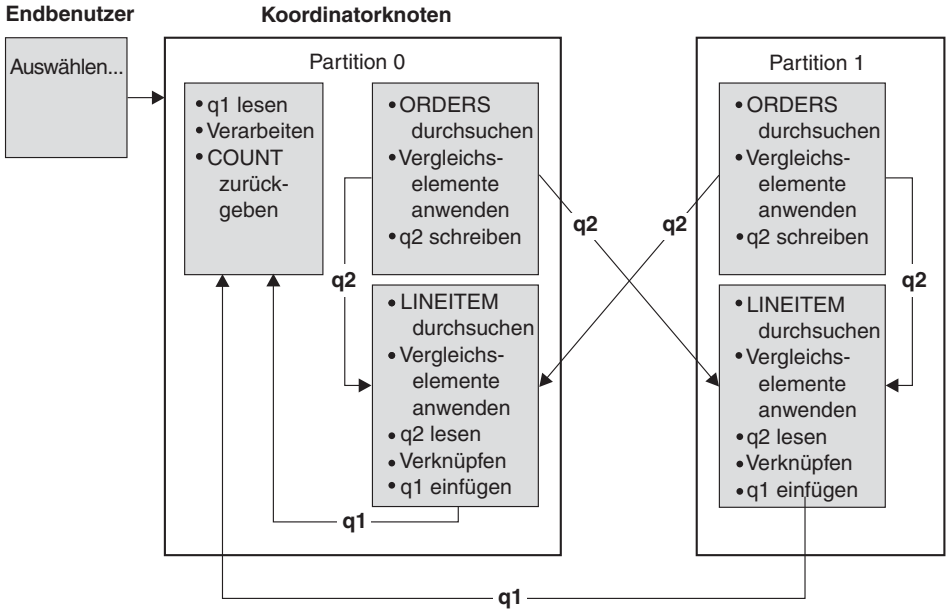


Die beiden Tabellen LINEITEM und ORDERS werden über die Spalte ORDERKEY partitioniert. Die Verknüpfung erfolgt lokal in jeder Datenbankpartition. In diesem Beispiel wird folgendes Vergleichselement für die Verknüpfung angenommen:
ORDERS.ORDERKEY = LINEITEM.ORDERKEY.

Abbildung 13. Beispiel für eine zusammengefasste Verknüpfung

Verknüpfungen mit rundgesendeter äußerer Tabelle

Verknüpfungen mit rundgesendeter äußerer Tabelle sind eine parallele Verknüpfungsstrategie, die angewandt werden kann, wenn es zwischen den zu verknüpfenden Tabellen keine Gleichheitsverknüpfungsprädikate gibt. Sie kann außerdem in solchen Fällen verwendet werden, in denen sie die Methode mit dem geringsten Aufwand darstellt. Zum Beispiel könnte eine Verknüpfung mit rundgesendeter äußerer Tabelle stattfinden, wenn eine sehr umfangreiche und eine sehr kleine Tabelle an der Verknüpfung beteiligt sind, von denen keine über die Spalten, auf die die Vergleichselemente angewandt werden, partitioniert ist. Anstatt beide Tabellen auf Partitionen zu verteilen, kann es „billiger" sein, die kleinere Tabelle an alle Partitionen mit der größeren Tabelle per Broadcast rundzusenden. Die folgenden Abbildungen geben ein Beispiel.

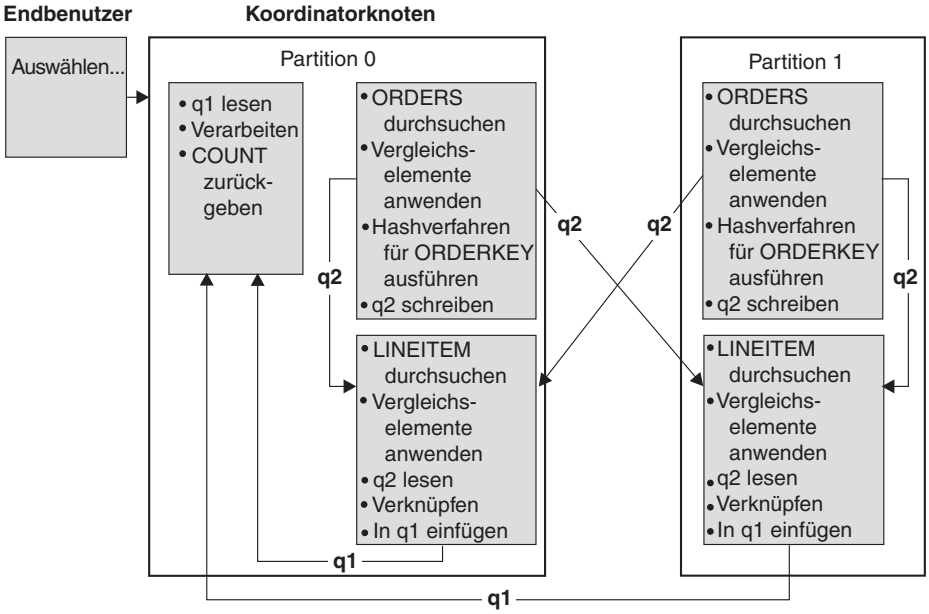


Die Tabelle ORDERS wird an alle Datenbankpartitionen mit der Tabelle LINEITEM gesendet. Tabellenwarteschlange q2 wird im Rundsendebetrieb an alle Datenbankpartitionen der inneren Tabelle gesendet.

Abbildung 14. Beispiel für eine Verknüpfung mit rundgesendeter äußerer Tabelle

Verknüpfungen mit gezielt übertragener äußerer Tabelle

Bei der Verknüpfungsstrategie mit gezielt übertragener äußerer Tabelle wird jede Zeile der äußeren Tabelle an eine Partition der inneren Tabelle entsprechend den Partitionierungsattributen der inneren Tabelle gesendet. Die Verknüpfung erfolgt in dieser Datenbankpartition. Die folgende Abbildung zeigt ein Beispiel.

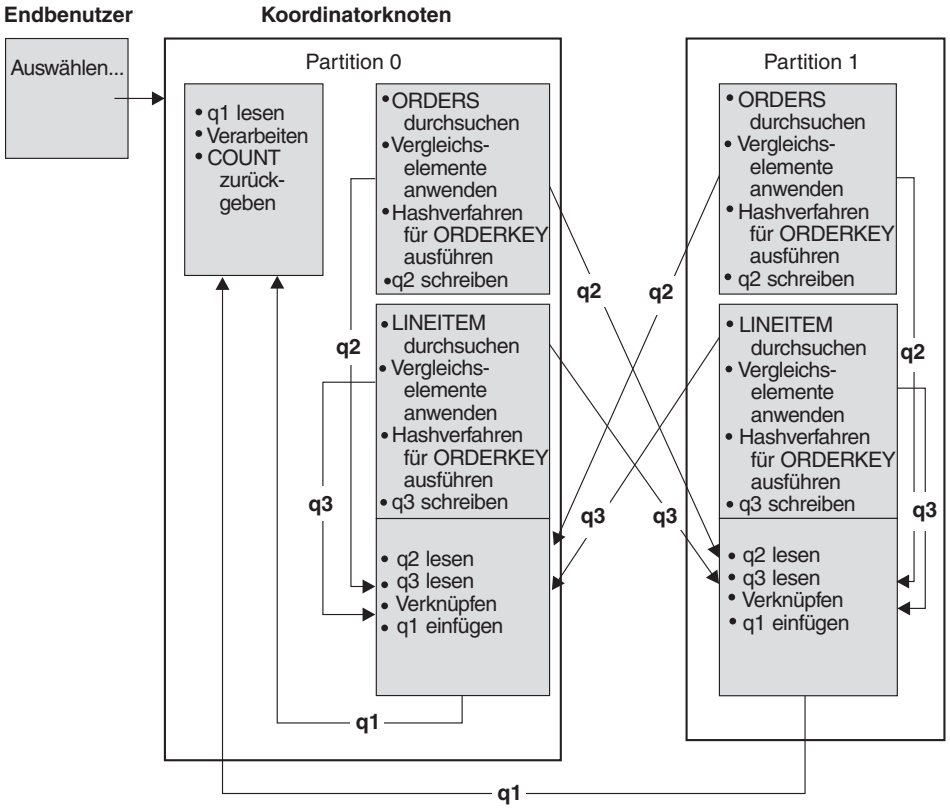


Die Tabelle LINEITEM wird über die Spalte ORDERKEY partitioniert.
Die Tabelle ORDERS wird über eine andere Spalte partitioniert.
Für die Tabelle ORDERS wird das Hashverfahren ausgeführt, und sie
wird an die richtige Datenbankpartition der Tabelle LINEITEM gesendet.
In diesem Beispiel wird folgendes Vergleichselement für
die Verknüpfung angenommen:
ORDERS.ORDERKEY = LINEITEM.ORDERKEY.

Abbildung 15. Beispiel für eine Verknüpfung mit gezielt übertragener äußerer Tabelle

Verknüpfungen mit gezielt übertragener innerer und äußerer Tabelle

Der Verknüpfungsstrategie mit gezielt übertragener innerer und äußerer Tabelle werden Zeilen sowohl der äußeren als auch der inneren Tabelle gezielt an eine Gruppe von Datenbankpartitionen entsprechend den Werten der Verknüpfungsspalten übertragen. Die Verknüpfung erfolgt in diesen Datenbankpartitionen. Die folgende Abbildung zeigt ein Beispiel.



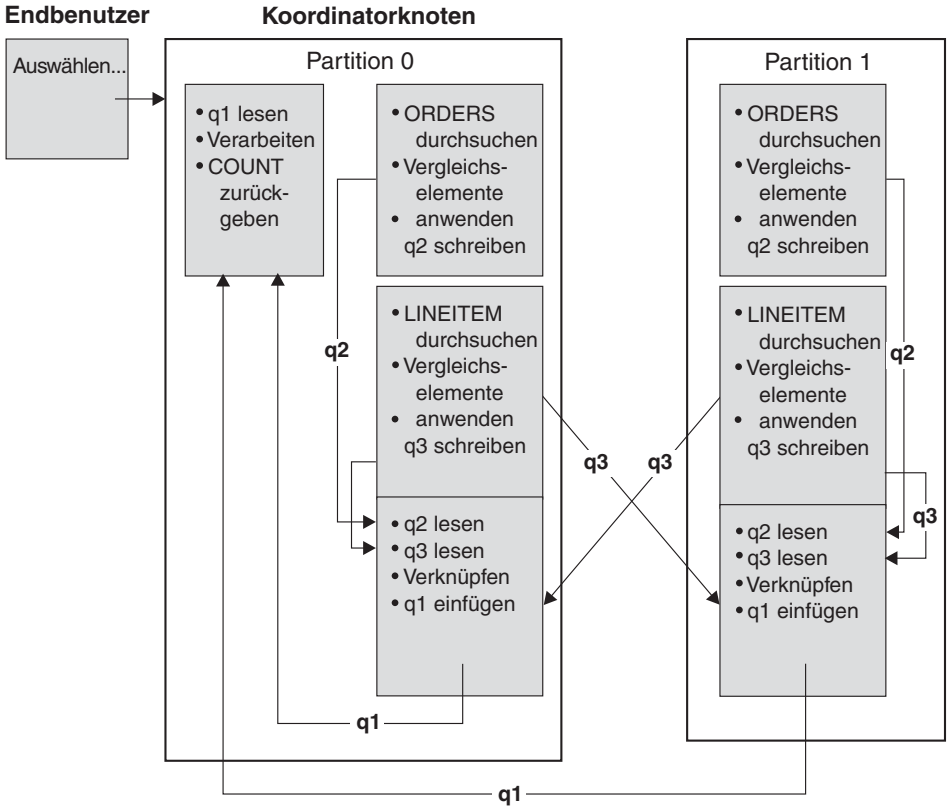
Es wird keine Tabelle über die Spalte ORDERKEY partitioniert.
Für beide Tabellen wird das Hashverfahren ausgeführt, und sie werden an neue Datenbankpartitionen gesendet und dort verknüpft.
Beide Tabellenwarteschlangen q2 und q3 werden übertragen.
In diesem Beispiel wird folgendes Vergleichselement für die Verknüpfung angenommen:

ORDERS.ORDERKEY = LINEITEM.ORDERKEY

Abbildung 16. Beispiel für eine Verknüpfung mit gezielt übertragener innerer und äußerer Tabelle

Verknüpfungen mit rundgesendeter innerer Tabelle

Bei der Verknüpfungsstrategie mit rundgesendeter innerer Tabelle wird die innere Tabelle per Broadcast an alle Datenbankpartitionen der äußeren Verknüpfungstabelle gesendet. Die folgende Abbildung zeigt ein Beispiel.

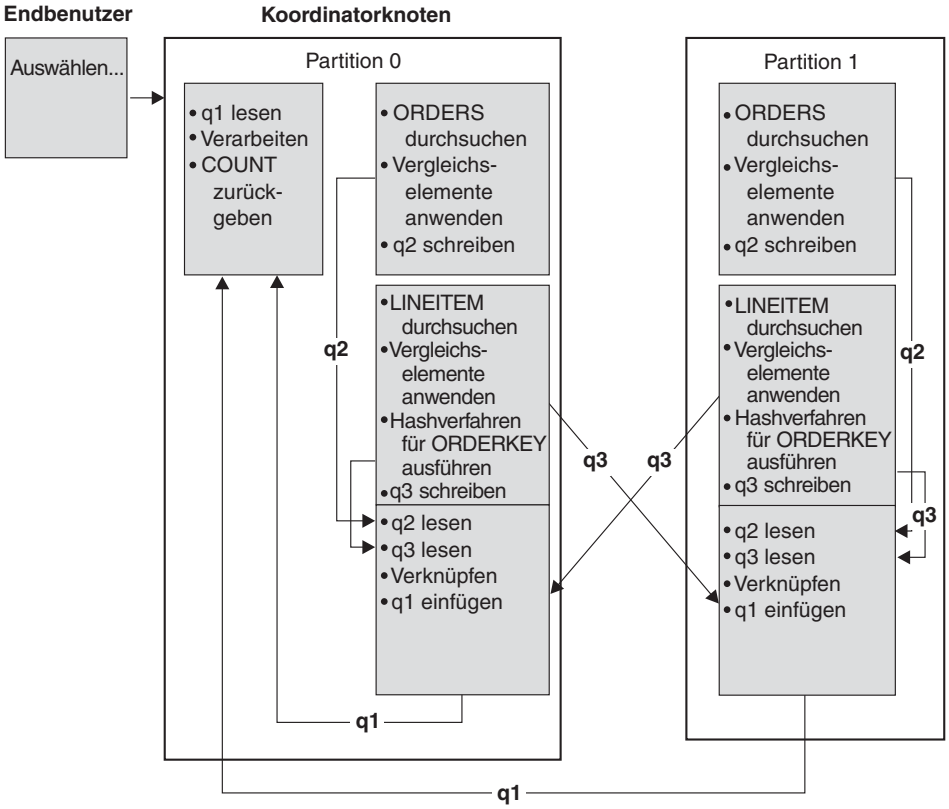


Die Tabelle LINEITEM wird an alle Datenbankpartitionen mit der Tabelle ORDERS gesendet. Tabellenwarteschlange q3 wird im Rundsendebetrieb an alle Datenbankpartitionen der äußeren Tabelle gesendet.

Abbildung 17. Beispiel für eine Verknüpfung mit rundgesendeter innerer Tabelle

Verknüpfungen mit gezielt übertragener innerer Tabelle

Bei der Verknüpfungsstrategie mit gezielt übertragener innerer Tabelle wird jede Zeile der inneren Tabelle an eine Datenbankpartition der äußeren Tabelle entsprechend den Partitionierungsattributen der äußeren Tabelle übertragen. Die Verknüpfung erfolgt in dieser Datenbankpartition. Die folgende Abbildung zeigt ein Beispiel.



Die Tabelle ORDERS wird über die Spalte ORDERKEY partitioniert.
Die Tabelle LINEITEM wird über eine andere Spalte partitioniert.
Für die Tabelle LINEITEM wird das Hashverfahren ausgeführt, und sie wird an die richtige Datenbankpartition der Tabelle ORDERS gesendet. In diesem Beispiel wird folgendes Vergleichselement für die Verknüpfung angenommen:
ORDERS.ORDERKEY = LINEITEM.ORDERKEY.

Abbildung 18. Beispiel für eine Verknüpfung mit gezielt übertragener innerer Tabelle

Zugehörige Konzepte:

- „Verknüpfungen" auf Seite 184
- „Verknüpfungsmethoden" auf Seite 185
- „Verknüpfungsstrategien in partitionierten Datenbanken" auf Seite 193

Auswirkungen des Sortierens und Gruppierens

Wenn das Optimierungsprogramm einen Zugriffsplan auswählt, kalkuliert es die Auswirkungen einer Sortierung von Daten auf die Leistung mit ein. Sortieroperationen werden durchgeführt, wenn kein Index die angeforderte Reihenfolge der abgerufenen Daten herstellen kann. Eine Sortierung kann auch erfolgen, wenn das Optimierungsprogramm feststellt, dass eine Sortierung weniger aufwendig als eine Indexsuche ist. Das Optimierungsprogramm sortiert Daten auf eine der folgenden Arten:

- Weitergeben der Sortierergebnisse über eine Pipe (Piping), wenn die Abfrage ausgeführt wird
- Interne Behandlung der Sortierung innerhalb des Datenbankmanagers

Vergleich von Sortierungen mit und ohne Piping

Wenn die endgültige, sortierte Liste von Daten in einem einzigen sequenziellen Vorgang gelesen werden kann, können die Ergebnisse über eine *Pipe* geleitet werden (Piping). Mit der Piping-Methode lassen sich die Ergebnisse der Sortierung schneller übertragen als mit Methoden ohne Piping. Das Optimierungsprogramm wählt, wenn möglich, die Pipe zur Übergabe der Sortierergebnisse.

Ungeachtet dessen, ob eine Sortierung mit oder ohne Piping erfolgt, hängt die für die Sortierung benötigte Zeit von einer Reihe von Faktoren ab, wie zum Beispiel der Anzahl der zu sortierenden Zeilen, der Größe des Sortierschlüssels und der Zeilenlänge. Wenn die zu sortierenden Zeilen mehr als den im Zwischenspeicher für Sortierlisten 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 Tabelle im Pufferpool gespeichert. Falls im Pufferpool nicht genügend Platz vorhanden ist, können Seiten aus dieser temporären Tabelle auf die Platte geschrieben werden. Wenn alle Arbeitsgänge des Sortiervorgangs abgeschlossen sind, müssen die sortierten Untermengen zu einer einzigen sortierten Menge von Zeilen zusammengefügt werden. Wenn die Sortierung über eine Pipe geleitet wird, werden die Zeilen beim Zusammenfügen direkt an die Services für relationale Daten (Relational Data Services) weitergegeben.

Pushdown von Gruppier- und Sortieroperatoren

In einigen Fällen kann sich das Optimierungsprogramm entscheiden, einen Sortiervorgang oder eine Spaltenberechnung (Aggregation) von der Komponente der Services für relationale Daten an die Komponente der Datenverwaltungsservices zu verschieben („Pushdown"). Eine solche Verschiebung dieser Operationen verbessert die Leistung, da nun die Datenverwaltungsservices Daten direkt an eine Sortier- oder Spaltenberechnungsroutine übergeben können. Ohne diese Verschiebung übergeben die Datenverwaltungsservices diese Daten zunächst an die Services für relationale Daten, die anschließend wiederum mit den Sortier- bzw. Spaltenberechnungsroutinen kommunizieren. Die folgende Abfrage beispielsweise kann von dieser Art der Optimierung profitieren:

```
SELECT WORKDEPT, AVG(SALARY) AS AVG_DEPT_SALARY
  FROM EMPLOYEE
  GROUP BY WORKDEPT
```

Gruppierungsoperationen in Sortierungen

Wenn das Sortieren dazu dient, die erforderliche Reihenfolge für eine Operation GROUP BY herzustellen, kann das Optimierungsprogramm einige oder alle Spaltenberechnungen (Aggregation) für GROUP BY während des Sortierens durchführen. Dies ist vorteilhaft, wenn die Anzahl der Zeilen in jeder Gruppe sehr groß ist. Es wird sogar noch vorteilhafter, wenn die Durchführung eines Teils der Gruppierung während des Sortierens die Notwendigkeit, dass die Sortierung einen Überlauf auf die Festplatte verursacht, verringert oder ausschließt.

Eine Spaltenberechnung in einer Sortierung erfordert unter Umständen alle der drei folgenden Phasen der Spaltenberechnung, um sicherzustellen, dass die richtigen Ergebnisse zurückgegeben werden.

1. Die erste Phase der Spaltenberechnung, die partielle Spaltenberechnung, errechnet die Ergebniswerte, bis der Sortierspeicher voll ist. Bei der partiellen Spaltenberechnung werden nicht berechnete Daten entgegengenommen und partielle Ergebniswerte erstellt. Wenn der Sortierspeicher voll ist, läuft der Rest der Daten auf die Festplatte über, einschließlich aller partiellen Spaltenberechnungsergebnisse, die im aktuellen Sortierspeicher berechnet wurden. Nach dem Zurücksetzen des Sortierspeichers werden neue Spaltenberechnungen gestartet.
2. Die zweite Phase der Spaltenberechnung, die Zwischenberechnung, nimmt alle übergelaufenen Sortierdurchläufe und setzt die Spaltenberechnung für die Gruppierschlüssel fort. Die Spaltenberechnung kann nicht zu Ende geführt werden, weil die Gruppierschlüsselspalten eine Untergruppe der Partitionierungsschlüsselspalten sind. Die Zwischenberechnung stellt aus vorhandenen partiellen Spaltenberechnungen neue partielle Spaltenberechnungen. Diese Phase findet nicht immer statt. Sie wird sowohl für partitionsinterne als auch für partitionsübergreifende Parallelität verwendet. Bei der partitionsinternen Parallelität ist die Gruppierung beendet, wenn ein globaler Gruppierschlüssel verfügbar ist. Bei partitionsübergreifender Parallelität findet sie statt, wenn der Gruppierschlüssel eine Untergruppe des Partitionierungsschlüssels ist, der Gruppen auf Partitionen verteilt, und so eine erneute Partitionierung zur Beendigung der Spaltenberechnung erforderlich macht. Ein ähnlicher Fall liegt vor, wenn bei partitionsinterner Parallelität jeder Agent seine übergelaufenen Sortierdurchgänge zusammengefügt hat, bevor auf einen einzigen Agenten reduziert wird, um die Spaltenberechnung zu beenden.
3. Die letzte Phase der Spaltenberechnung, die Endberechnung, verwendet alle partiellen Berechnungsergebnisse und erstellt die endgültige Spaltenberechnung. Dieser Schritt findet immer in einem Operator GROUP BY statt. Das Sortieren kann die Spaltenberechnung nicht bis zu Ende durchführen, weil es keine Garantie gibt, dass die Sortierung nicht geteilt wird. Die Komplettberechnung nimmt nicht berechnete Daten auf und produziert Endergebnisse. Wenn die Partitionierung die Nutzung dieser Methode nicht verhindert, wird die Methode der Spaltenberechnung in der Regel zur Gruppierung von Daten verwendet, die bereits in der richtigen Reihenfolge vorliegen.

Zugehörige Konzepte:

- „Richtlinien für die Sortierleistung" auf Seite 278

Zugehörige Referenzen:

- „sortheap - Zwischenspeichergröße für Sortierlisten" auf Seite 418
- „sheapthres - Schwellenwert für Sortierspeicher" auf Seite 416

Optimierungsstrategien

Dieser Abschnitt beschreibt die besonderen Strategien, die das Optimierungsprogramm für die partitionsinterne Parallelität und für MDC-Tabellen (Multidimensional Clustering) anwenden kann.

Optimierungsstrategien für partitionsinterne Parallelität

Das Optimierungsprogramm kann einen Zugriffsplan wählen, der eine Abfrage parallel innerhalb einer Datenbankpartition ausführt, wenn ein Grad von Parallelität bei der Kompilierung der SQL-Anweisung angegeben wird.

Während der Ausführung werden mehrere Datenbankagenten, so genannte Subagenten, zur Ausführung der Abfrage erstellt. Die Anzahl von Subagenten ist kleiner oder gleich dem Grad von Parallelität, der bei der Kompilierung der SQL-Anweisung angegeben wurde.

Zur Parallelisierung unterteilt das Optimierungsprogramm den Zugriffsplan in Teile, die jeweils von einem Subagenten, sowie in einen Teil, der vom koordinierenden Agenten ausgeführt wird. Die Subagenten übergeben Daten über Tabellenwarteschlangen an den koordinierenden Agenten oder andere Subagenten. In einer partitionierten Datenbank können Subagenten Daten an Subagenten in anderen Datenbankpartitionen senden oder von ihnen empfangen.

Strategien zur partitionsinternen Parallelsuche

Tabellensuchen und Indexsuchen können parallel in derselben Tabelle oder demselben Index ausgeführt werden. Für parallele Tabellensuchen wird die Tabelle in Seiten- oder Zeilenbereiche unterteilt. Je ein Bereich von Seiten oder Zeilen wird einem Subagenten zugewiesen. Ein Subagent durchsucht den ihm zugewiesenen Bereich und erhält einen anderen Bereich zugewiesen, wenn er mit dem Durchsuchen des aktuellen Bereichs fertig ist.

Für parallele Indexsuchen wird der Index in Bereiche von Datensätzen entsprechend den Indexschlüsselwerten und der Anzahl von Indexeinträgen für einen Schlüsselwert unterteilt. Die parallele Indexsuche wird wie die parallele Tabellensuche mit Subagenten durchgeführt, denen jeweils ein Bereich von Datensätzen zugewiesen wird. Einem Subagenten wird ein neuer Bereich zugewiesen, wenn er die Suche im aktuellen Bereich beendet hat.

Das Optimierungsprogramm legt die Sucheinheit (entweder Seite oder Zeile) sowie die Suchgranularität fest.

Bei Parallelsuchen wird die Arbeit gleichmäßig unter den Subagenten verteilt. Der Zweck einer Parallelsuche besteht darin, die Belastung unter den Subagenten ausgewogen zu verteilen und sie gleichmäßig auszulasten. Wenn die Anzahl aktiver Subagenten gleich der Anzahl verfügbarer Prozessoren ist und die Platten nicht mit E/A-Anforderungen überlastet sind, werden die Ressourcen der Maschine effektiv genutzt.

Andere Zugriffsplanstrategien können eine unausgewogene Datenverteilung bei der Ausführung der Abfrage verursachen. Das Optimierungsprogramm wählt Parallelstrategien aus, die für eine ausgewogene Datenverteilung unter den Subagenten sorgen.

Strategien zur partitionsinternen parallelen Sortierung

Das Optimierungsprogramm kann eine der folgenden parallelen Sortierstrategien auswählen:

- **Reihumsortierung**

  Dies kann auch als *Umverteilungssortieren* bezeichnet werden. Diese Methode nutzt den gemeinsamen Speicher effizient, um die Daten so gleichmäßig wie möglich auf alle Subagenten zu verteilen. Zur Realisierung der gleichmäßigen Verteilung wird eine Art Reihumverteilungsalgorithmus verwendet. Zunächst wird ein Sortiervorgang für jeden Subagenten erstellt. Während der Einfügephase fügen Subagenten Zeilen reihum in jeden der einzelnen Sortiervorgänge ein, um eine gleichmäßigere Datenverteilung zu erzielen.
- **Partitionierte Sortierung**

  Dies ist dem reihumverteilten Sortieren insofern ähnlich, als dass für jeden Subagenten ein Sortiervorgang erstellt wird. Die Subagenten wenden eine Hashfunktion auf die Sortierspalten an, um festzulegen, in welchen Sortiervorgang eine Zeile einzufügen ist. Wenn beispielsweise die innere und die äußere Tabelle einer Mischverknüpfung in einem partitionierten Sortiervorgang sind, kann ein Subagent mit Hilfe einer Mischverknüpfung die entsprechenden Partitionen verknüpfen und parallel ausgeführt werden.
- **Replizierte Sortierung**

  Diese Art der Sortierung wird verwendet, wenn jeder Subagent die gesamte Ausgabe der Sortierung benötigt. Es wird nur ein Sortiervorgang erstellt. Beim Einfügen von Zeilen in den Sortiervorgang werden die Subagenten synchronisiert. Nach Beendigung der Sortierung liest jeder Subagent das gesamte Sortierergebnis. Wenn die Anzahl von Zeilen gering ist, kann diese Art der Sortierung zum erneuten Ausgleich des Datenstroms verwendet werden.
- **Gemeinsame Sortierung**

  Diese Art der Sortierung entspricht der replizierten Sortierung, abgesehen davon, dass die Subagenten eine parallele Suche über das Sortierergebnis öffnen, um die Daten unter den Subagenten ähnlich wie bei einer Reihumsortierung zu verteilen.

Partitionsinterne temporäre Paralleltabellen

Subagenten können kooperieren, um eine temporäre Tabelle durch Einfügen von Zeilen in dieselbe Tabelle zu erstellen. Eine solche Tabelle ist eine `gemeinsame temporäre Tabelle`. Die Subagenten können private oder parallele Suchoperationen über die gemeinsame temporäre Tabelle öffnen, je nachdem, ob der Datenstrom zu replizieren oder zu partitionieren ist.

Strategien zur partitionsinternen parallelen Spaltenberechnung

Spaltenberechnungen (Aggregation) können von Subagenten parallel durchgeführt werden. Eine Spaltenberechnung setzt voraus, dass die Daten nach den Gruppierungsspalten geordnet sind. Wenn ein Subagent sicher sein kann, dass er alle Zeilen für eine Reihe von Gruppierungsspaltenwerten erhält, kann er eine vollständige Spaltenberechnung (Aggregation) durchführen. Dies ist möglich, wenn der Strom bereits aufgrund einer früheren partitionierten Sortierung über die Gruppierungsspalten partitioniert ist.

Andernfalls kann der Subagent eine partielle Spaltenberechnung durchführen und eine andere Strategie zur Vervollständigung der Spaltenberechnung anwenden. Einige dieser Strategien sind:

- Senden der teilweise berechneten Daten an den Koordinatoragenten über eine Tabellenwarteschlange für Mischverknüpfungen. Der Koordinatoragent vervollständigt die Spaltenberechnung.
- Einfügen der teilweise berechneten Daten in eine partitionierte Sortierung. Die Sortierung wird über die Gruppierungsspalten partitioniert und stellt sicher, dass alle Zeilen für eine Reihe von Gruppierungsspalten in einer Sortierpartition enthalten sind.
- Wenn der Strom zum Ausgleichen der Verarbeitung repliziert werden muss, können die teilweise berechneten Daten in eine replizierte Sortierung eingefügt werden. Die einzelnen Subagenten vervollständigen die Spaltenberechnung über die replizierte Sortierung und erhalten eine identische Kopie des Ergebnisses der Spaltenberechnung.

Strategien zur partitionsinternen parallelen Verknüpfung

Verknüpfungsoperationen können von Subagenten parallel durchgeführt werden. Parallele Verknüpfungsstrategien werden durch die Merkmale des Datenstroms festgelegt.

Eine Verknüpfung kann parallelisiert werden, indem der Datenstrom nach der inneren und äußeren Tabelle der Verknüpfung partitioniert, repliziert oder beides wird. Zum Beispiel kann eine Verknüpfung über Verschachtelungsschleife parallelisiert werden, wenn der äußere Datenstrom für eine Parallelsuche partitioniert ist und der innere Datenstrom von jedem Subagenten unabhängig neu ausgewertet wird. Eine Mischverknüpfung kann parallelisiert werden, wenn der innere und der äußere Datenstrom für partitionierte Sortierungen nach ihren Werten partitioniert sind.

Zugehörige Konzepte:

- „Parallelverarbeitung für Anwendungen" auf Seite 103
- „Optimierungsstrategien für MDC-Tabellen" auf Seite 206

Optimierungsstrategien für MDC-Tabellen

Wenn Sie Tabellen mit mehrdimensionalem Clustering (MDC - Multidimensional Clustering) erstellen, kann sich die Leistung vieler Abfragen verbessern, weil das Optimierungsprogramm zusätzliche Optimierungsstrategien anwenden kann. Diese Strategien basieren in erster Linie auf der verbesserten Effizienz von Blockindizes, jedoch ermöglicht der Vorteil des mehrdimensionalen Clustering auch einen schnelleren Datenabruf.

Anmerkung: Die Optimierungsstrategien für MDC-Tabellen können auch die Leistungsvorteile der partitionsinternen und partitionsübergreifenden Parallelität implementieren.

MDC-Tabellen bieten die folgenden spezifischen Vorteile:

- Dimensionsblockindexsuchen können die erforderlichen Teile der Tabelle ermitteln und schnell nur die angeforderten Blöcke durchsuchen.
- Da Blockindizes kleiner als Satz-ID-Indizes sind, arbeiten Blockindexsuchen schneller.
- AND- und OR-Verknüpfungen von Indizes (Index ANDing und Index ORing) können auf Blockebene durchgeführt und mit Satz-IDs kombiniert werden.
- Daten werden garantiert in EXTENTSIZE-Speicherbereichen in Clustern gruppiert, was ein schnelleres Abrufen ermöglicht.

Betrachten Sie das folgende einfache Beispiel für eine MDC-Tabelle mit dem Namen **sales**, in der Dimensionen auf den Spalten **region** und **month** definiert sind:

```
SELECT * FROM SALES
    WHERE MONTH='March' AND REGION='SE'
```

Für diese Abfrage kann das Optimierungsprogramm eine Dimensionsblockindexsuche durchführen, um die Blöcke zu finden, in denen der Monat März (March) und die Region SE vorkommen. Anschließend kann es nur die resultierenden Blöcke der Tabelle durchsuchen, um die Ergebnismenge abzurufen.

Zugehörige Konzepte:

- „Tabellen- und Indexverwaltung für MDC-Tabellen" auf Seite 24

Gespeicherte Abfragetabellen

Gespeicherte Abfragetabellen (MQTs) bieten eine leistungsstarke Möglichkeit, die Antwortzeit für komplexe Abfragen zu verbessern, insbesondere für Abfragen, für die einige der folgenden Operationen erforderlich sind:

- Erstellen von Ergebnisdaten für eine oder mehrere Dimensionen
- Verknüpfungen und Spaltenberechnungen für eine Gruppe von Tabellen
- Bereitstellen von Daten aus einer häufig genutzten Untergruppe von Daten, d. h. einer sofort für Verarbeitungsoperationen bereiten horizontalen oder vertikalen Partition
- Erneutes Partitionieren von Daten aus einer Tabelle bzw. einem Teil einer Tabelle in einer partitionierten Datenbankumgebung

In den SQL-Compiler sind Kenntnisse über gespeicherte Abfragetabellen integriert. Im SQL-Compiler werden in der Phase des Umschreibens von Abfragen und durch das Optimierungsprogramm Abfragen mit gespeicherten Abfragetabellen verglichen, um zu ermitteln, ob anstelle einer Abfrage, die auf die Basistabellen zugreift, eine gespeicherte Abfragetabelle verwendet werden kann. Wenn eine gespeicherte Abfragetabelle verwendet wird, kann die EXPLAIN-Einrichtung Informationen darüber bereitstellen, welche gespeicherte Abfragetabelle ausgewählt wurde.

Da sich gespeicherte Abfragetabellen in vielerlei Hinsicht wie reguläre Tabellen verhalten, treffen die Richtlinien zum Optimieren des Datenzugriffs unter Verwendung von Tabellenbereichsdefinitionen, durch Erstellen von Indizes und Ausführen des Dienstprogramms RUNSTATS auch auf gespeicherte Abfragetabellen zu.

Zur Veranschaulichung der Leistungsfähigkeit gespeicherter Abfragetabellen wird im folgenden Beispiel eine mehrdimensionale Analyseabfrage dargestellt und erläutert, wie sie von gespeicherten Abfragetabellen profitiert.

Nehmen Sie für dieses Beispiel ein Datenbankschema an, in dem ein Data Warehouse eine Reihe von Kunden und eine Reihe von Kreditkartenkonten enthält. Das Data Warehouse zeichnet die Gruppe der Transaktionen auf, die mit den Kreditkarten durchgeführt wurden. Alle Transaktionen enthalten eine Anzahl von Artikeln, die gemeinsam gekauft wurden. Dieses Schema wird als Mehrfachsternschema (Multi-Star) eingestuft, da zwei große Tabellen, von denen die eine Transaktionselemente enthält und die andere die Kauftransaktionen identifiziert, zusammen das Zentrum des Sterns bilden.

Die drei hierarchischen Dimensionen, die eine Transaktion beschreiben, sind Produkt, Standort und Zeit. Die Produkthierarchie wird in zwei normalisierten Tabellen gespeichert, die die Produktgruppe und die Produktlinie darstellen. Die Standorthierarchie enthält Informationen zu Ort, Bundesland, sowie Land oder Region, die in einer einzigen denormalisierten Tabelle dargestellt werden. Die Zeithierarchie enthält Informationen zu Tag, Monat und Jahr und ist in einem einzigen Datumsfeld codiert. Die Datumsdimensionen werden aus dem Datumsfeld der Transaktion unter Verwendung integrierter Funktionen extrahiert. Andere Tabellen in diesem Schema stellen die Kontoinformationen für Kunden und Kundeninformationen dar.

Eine gespeicherte Abfragetabelle wird mit der Summe und der Anzahl der Verkäufe für jede Stufe der folgenden Hierarchien erstellt:

- Produkt
- Standort
- Zeit bestehend aus Jahr, Monat, Tag

Viele Abfragen können aus diesen zusammengefassten Ergebnisdaten erfüllt werden. Das folgende Beispiel zeigt, wie eine gespeicherte Abfrage erstellt wird, die die Summe und die Anzahl der Verkäufe für die Dimensionen 'Produktgruppe' (Product Group) und 'Produktlinie (Product Line), für die Dimensionen 'Ort' (City), 'Bundesland' (State) und 'Land' (Country) und für die Dimension 'Zeit' (Time) berechnet. Das Beispiel enthält auch einige weitere Spalten in der Klausel GROUP BY.

```
CREATE TABLE dba.PG_SALESSUM
  AS (
      SELECT l.id AS prodline, pg.id AS pgroup,
             loc.country, loc.state, loc.city,
             l.name AS linename, pg.name AS pgname,
             YEAR(pdate) AS year, MONTH(pdate) AS month,
             t.status,
             SUM(ti.amount) AS amount,
             COUNT(*) AS count
      FROM   cube.transitem AS ti, cube.trans AS t,
             cube.loc AS loc, cube.pgroup AS pg,
             cube.prodline AS l
      WHERE  ti.transid = t.id
         AND ti.pgid = pg.id
         AND pg.lineid = l.id
         AND t.locid = loc.id
         AND YEAR(pdate) > 1990
      GROUP BY l.id, pg.id, loc.country, loc.state, loc.city,
               year(pdate), month(pdate), t.status, l.name, pg.name
     )
DATA INITIALLY DEFERRED REFRESH DEFERRED;

REFRESH TABLE dba.SALESCUBE;
```

Abfragen, die solche vorberechneten Summen nutzen können, sind unter anderem:
- Verkaufsdaten nach Monat und Produktgruppe
- Gesamtvertrieb für Jahre nach 1990
- Verkauf für 1995 oder 1996
- Summe des Verkaufs für eine Produktgruppe oder Produktlinie
- Summe des Verkaufs für eine bestimmte Produktgruppe oder Produktlinie UND für 1995, 1996
- Summe des Verkaufs für ein bestimmtes Land

Obwohl die präzise Anwort für keine dieser Abfragen in der gespeicherten Abfragetabelle enthalten ist, könnte der Aufwand zur Berechnung der Antwort mit Hilfe der gespeicherten Abfragetabelle erheblich geringer ausfallen als bei Verwendung der umfangreichen Basistabelle, da ein Teil der für die Antwort benötigten Berechnungen bereits erfolgt ist. Gespeicherte Abfragetabellen können die Notwendigkeit von aufwendigen Verknüpfungen, Sortierungen und Spaltenberechnungen von Basisdaten verringern.

Die folgenden Beispielabfragen könnten beträchtliche Leistungsverbesserungen erfahren, da sie die bereits berechneten Ergebnisse der gespeicherten Beispielabfragetabelle nutzen könnten.

Das erste Beispiel liefert die Gesamtverkäufe für 1995 und 1996:

```
SET CURRENT REFRESH AGE=ANY

SELECT YEAR(pdate) AS year, SUM(ti.amount) AS amount
FROM   cube.transitem AS ti, cube.trans AS t,
       cube.loc AS loc, cube.pgroup AS pg,
       cube.prodline AS l
WHERE  ti.transid = t.id
```

```
   AND ti.pgid = pg.id
   AND pg.lineid = l.id
   AND t.locid = loc.id
   AND YEAR(pdate) IN (1995, 1996)
GROUP BY year(pdate);
```

Das zweite Beispiel liefert die Gesamtverkäufe nach Produktgruppe für 1995 und 1996:

```
SET CURRENT REFRESH AGE=ANY

SELECT pg.id AS "PRODUCT GROUP",
       SUM(ti.amount) AS amount
FROM   cube.transitem AS ti, cube.trans AS t,
       cube.loc AS loc, cube.pgroup AS pg,
       cube.prodline AS l
WHERE  ti.transid = t.id
   AND ti.pgid = pg.id
   AND pg.lineid = l.id
   AND t.locid = loc.id
   AND YEAR(pdate) IN (1995, 1996)
GROUP BY pg.id;
```

Je größer die Basistabellen sind, desto höher können die Leistungsverbesserungen in Antwortzeiten ausfallen, weil die gespeicherte Abfragetabelle langsamer anwächst als die Basistabelle. Gespeicherte Abfragetabellen können sich überlappende Arbeitsschritte von Abfragen effektiv vermeiden helfen, indem sie die Berechnungen einmal bei ihrer Erstellung und Aktualisierung (REFRESH) durchführen und ihren Inhalt anschließend vielen Abfragen zur Verfügung stellen.

Zugehörige Konzepte:

- „Der Designadvisor" auf Seite 237
- „Replizierte gespeicherte Abfragetabellen in partitionierten Datenbanken" auf Seite 191

Compilerphasen für Abfragen zusammengeschlossener Datenbanken

In diesem Abschnitt werden die zusätzlichen Phasen der Abfrageverarbeitung in einem System zusammengeschlossener Datenbanken beschrieben. Es enthält darüber hinaus Empfehlungen für die Verbesserung der Leistung von Abfragen für zusammengeschlossene Datenbanken.

Pushdown-Analyse für zusammengeschlossene Datenbanken

Für Abfragen in zusammengeschlossenen Datenbanken führt das Optimierungsprogramm eine Pushdown-Analyse durch, um zu ermitteln, ob eine Operation an einer fernen Datenquelle durchgeführt werden kann. Bei dieser Operation kann es sich um eine Funktion, wie zum Beispiel einen relationalen Operator, eine System- oder Benutzerfunktion oder einen SQL-Operator, zum Beispiel GROUP BY, ORDER BY usw., handeln.

Anmerkung: Obwohl der SQL-Compiler von DB2® über zahlreiche Informationen zur SQL-Unterstützung der Datenquelle verfügt, müssen diese Daten eventuell mit der Zeit angepasst werden, weil Datenquellen erweitert und/oder angepasst werden können. In solchen Fällen müssen Erweiterungen DB2 durch Ändern der lokalen Kataloginformationen bekannt gemacht werden. Verwenden Sie DDL-Anweisungen von DB2 (z. B. CREATE FUNCTION MAPPING und ALTER SERVER), um den Katalog zu aktualisieren.

Wenn Funktionen nicht an die ferne Datenquelle verschoben werden können, können sie sich erheblich auf die Abfrageleistung auswirken. Betrachten Sie die Konsequenzen für den Fall, dass ein selektives Vergleichselement lokal anstatt an der Datenquelle ausgewertet werden muss. Eine solche Auswertung würde DB2 zwingen, die gesamte Tabelle von der fernen Datenquelle abzurufen und anschließend lokal über das Vergleichselement zu filtern. Netzwerkbegrenzungen und eine bedeutende Tabellengröße könnten zu einer Beeinträchtigung der Leistung führen.

Operatoren, die nicht an die Datenquelle verschoben werden, können sich ebenfalls bedeutsam auf die Abfrageleistung auswirken. Wenn zum Beispiel ein Operator GROUP BY für ferne Daten lokal ausgeführt wird, muss DB2 ebenfalls die gesamte Tabelle von der fernen Datenquelle abrufen.

Nehmen Sie zum Beispiel an, dass der Kurzname N1 auf die Tabelle EMPLOYEE an einer Datenquelle unter DB2 für OS/390® oder z/OS verweist. Nehmen Sie weiter an, dass die Tabelle 10.000 Zeilen hat und dass eine der Spalten die Familiennamen (Lastname) und eine andere die Gehälter (Salary) enthält. Betrachten Sie die folgende Anweisung:

```
SELECT LASTNAME, COUNT(*)  FROM N1
   WHERE LASTNAME  > 'B' AND SALARY > 50000
   GROUP BY LASTNAME;
```

Es werden verschiedene Möglichkeiten in Betracht gezogen, je nachdem, ob die Sortierfolgen unter DB2 und DB2 für OS/390 oder z/OS übereinstimmen:

- Wenn die Sortierfolgen übereinstimmen, wird das Abfragevergleichselement wahrscheinlich an die Datenquelle unter DB2 für OS/390 oder z/OS verschoben (Pushdown). Eine Filterung und Gruppierung der Ergebnisse an der Datenquelle ist in der Regel effizienter, als die gesamte Tabelle nach DB2 zu kopieren und die Operationen lokal auszuführen. Für die oben gezeigte Abfrage können die Operationen für das Vergleichselement und für den Operator GROUP BY an der Datenquelle stattfinden.
- Wenn die Sortierfolgen nicht übereinstimmen, kann das gesamte Vergleichselement nicht an der Datenquelle ausgewertet werden. Allerdings kann das Optimierungsprogramm entscheiden, den Teil `SALARY > 50000` des Vergleichselements an die Datenquelle zu verschieben. Der Vergleich des Wertebereichs muss immer noch in DB2 ausgeführt werden.
- Wenn die Sortierfolgen übereinstimmen und dem Optimierungsprogramm bekannt ist, dass der lokale DB2-Server sehr schnell ist, kann das Optimierungsprogramm entscheiden, dass die lokale Ausführung der Operation `GROUP BY` in DB2 das günstigste Verfahren (d. h. das mit dem geringsten Aufwand) ist. Das Vergleichselement wird an der Datenquelle ausgeführt. Dies wäre ein Beispiel für die Pushdown-Analyse in Kombination mit globaler Optimierung.

Im Allgemeinen besteht das Ziel darin, sicherzustellen, dass das Optimierungsprogramm Funktionen und Operatoren an den Datenquellen auswertet. Zahlreiche Faktoren haben Einfluss auf die Entscheidung, ob eine Funktion oder ein SQL-Operator an der fernen Datenquelle ausgewertet wird. Die zu bewertenden Faktoren werden in folgende drei Gruppen klassifiziert:

- Servermerkmale
- Kurznamenmerkmale
- Abfragemerkmale

Servermerkmale mit Auswirkung auf die Pushdown-Möglichkeiten

Bestimmte datenquellenspezifische Faktoren können sich auf die Pushdown-Möglichkeiten auswirken. Diese Faktoren gibt es im Allgemeinen deswegen, weil DB2 eine sehr variantenreiche SQL-Version unterstützt. Diese SQL-Version bietet u. U. mehr Funktionalität als die SQL-Version, die von einem Server unterstützt wird, auf den durch eine Abfrage zugegriffen wird. DB2 kann diesen Funktionsmangel auf dem Datenserver kompensieren, allerdings kann dies dazu führen, dass die Operation in DB2 lokal ausgeführt werden muss.

SQL-Leistungsspektrum: Jede Datenquelle unterstützt eine Variante der SQL-Version und verschiedene Funktionalitätsebenen. Betrachten Sie zum Beispiel die GROUP BY-Liste. Die meisten Datenquellen unterstützen den Operator GROUP BY. Jedoch begrenzen einige die Anzahl von Elementen in der GROUP BY-Liste oder haben Einschränkungen hinsichtlich der Zulässigkeit bestimmter Ausdrücke in der GROUP BY-Liste. Wenn es eine Einschränkung an der fernen Datenquelle gibt, muss DB2 die Operation GROUP BY eventuell lokal ausführen.

SQL-Einschränkungen: Jede Datenquelle kann unterschiedliche SQL-Einschränkungen haben. Zum Beispiel fordern einige Datenquellen, dass Parametermarken Werte an ferne SQL-Anweisungen binden. Daher müssen die Einschränkungen für Parametermarken überprüft werden, um sicherzustellen, dass jede Datenquelle solche Bindeverfahren unterstützen kann. Wenn DB2 keine gute Methode zum Binden eines Werts für eine Funktion finden kann, muss diese Funktion lokal ausgewertet werden.

SQL-Begrenzungen: DB2 lässt vielleicht die Verwendung größerer ganzer Zahlen (Integer) als die fernen Datenquellen zu. Allerdings können Werte, die ferne Grenzwerte überschreiten, nicht in Anweisungen eingebettet werden, die an Datenquellen gesendet werden. Daher muss eine Funktion oder ein Operator, die bzw. der mit einer solchen Konstanten operiert, lokal ausgewertet werden.

Serverspezifische Faktoren: In diese Kategorie fallen verschiedene Faktoren. Ein Beispiel ist, ob NULL-Werte als höchster oder niedrigster Wert sortiert werden oder von der Reihenfolge abhängig sind. Wenn NULL-Werte an einer Datenquelle anders als von DB2 sortiert wird, können Operationen ORDER BY für einen Ausdruck mit möglichem Nullwert nicht fern ausgewertet werden.

Sortierfolge: Das Abrufen von Daten für lokale Sortierungen und Vergleiche setzt in der Regel die Leistung herab. Daher sollten Sie in Erwägung ziehen, die zusammengeschlossene Datenbank so zu konfigurieren, dass sie dieselbe Sortierfolge wie die Datenquellen verwendet. Wenn Sie eine zusammengeschlossene Datenbank zur Verwendung derselben Sortierfolge konfigurieren, die von einer Datenquelle verwendet wird, und die Serveroption *collating_sequence* auf den Wert 'Y' setzen, kann das Optimierungsprogramm in Betracht ziehen, viele Abfrageoperationen an die Datenquelle zu verschieben, wenn sich dadurch die Leistung verbessert.

Die folgenden Operationen können an eine Datenquelle verschoben werden, wenn die Sortierfolgen übereinstimmen:
- Vergleiche von Zeichendaten oder numerischen Daten
- Vergleichselemente für Zeichenbereiche
- Sortierungen

Unerwartete Ergebnisse kommen eventuell dann zustande, wenn die Wertigkeit von Nullzeichen zwischen der zusammengeschlossenen Datenbank und der Datenquelle unterschiedlich ist. Vergleichsanweisungen können unerwartete Ergebnisse

liefern, wenn Sie Anweisungen an eine Datenquelle übergeben, auf der die Groß-/Kleinschreibung nicht unterschieden wird. Die Wertigkeiten der Zeichen ″I″ und ″i″ sind bei einer Datenquelle ohne Unterscheidung der Groß-/Kleinschreibung identisch. DB2 beachtet standardmäßig die Groß-/Kleinschreibung und weist den Zeichen unterschiedliche Wertigkeiten zu.

Zur Verbesserung der Leistung lässt der Server der zusammengeschlossenen Datenbank zu, dass Sortierungen und Vergleiche an der Datenquelle stattfinden. Zum Beispiel werden in DB2 UDB für OS/390 oder z/OS Sortierungen, die durch Klauseln ORDER BY definiert werden, mit Hilfe einer Sortierfolge implementiert, die auf einer EBCDIC-Codepage (Extended Binary-Coded Decimal Interchange Code) basiert. Um den Server der zusammengeschlossenen Datenbank zum Abrufen von Daten einer Datenquelle unter DB2 für OS/390 oder z/OS zu verwenden und die Daten an der Datenquelle mit Hilfe von Klauseln ORDER BY zu sortieren, konfigurieren Sie die zusammengeschlossene Datenbank so, dass sie eine vordefinierte, auf der EBCDIC-Codepage basierende Sortierfolge verwendet.

Wenn die Sortierfolgen der zusammengeschlossenen Datenbank und der Datenquelle voneinander abweichen, ruft DB2 die Daten in die zusammengeschlossene Datenbank ab. Da Benutzer die Abfrageergebnisse nach der für den Server der zusammengeschlossenen Datenbank definierten Sortierfolge geordnet erwarten, stellt Server der zusammengeschlossenen Datenbank durch ein lokales Sortieren der Daten sicher, dass diese Erwartung erfüllt wird. Übergeben Sie Ihre Abfrage im Durchgriffsmodus (Pass-through) oder definieren Sie die Abfrage in einer Datenquellensicht, wenn Sie die Daten nach der Sortierfolge der Datenquelle geordnet abrufen müssen.

Serveroptionen: Verschiedene Serveroptionen können die Pushdown-Möglichkeiten beeinflussen. Prüfen Sie insbesondere Ihre Einstellungen für die Serveroptionen *collating_sequence*, *varchar_no_trailing_blanks* und *pushdown*.

Faktoren der DB2-Typenzuordnung und DB2-Funktionszuordnung: Die von DB2 definierten lokalen Standardtypenzuordnungen sind dazu vorgesehen, genügend Pufferspeicherplatz für jeden Datentyp einer Datenquelle bereitzustellen, wodurch Datenverlust vermieden wird. Benutzer können Typenzuordnung für eine bestimmte Datenquelle an die Anforderungen bestimmter Anwendungen anzupassen. Wenn Sie zum Beispiel auf eine Spalte des Datentyps DATE einer Oracle-Datenquelle zugreifen, die standardmäßig dem DB2-Datentyp TIMESTAMP zugeordnet wird, können Sie den lokalen Datentyp in den DB2-Datentyp DATE ändern.

In den folgenden drei Fällen kann DB2 Funktionen kompensieren, die von einer Datenquelle nicht unterstützt werden:

- Die Funktion ist an der fernen Datenquelle nicht vorhanden.
- Die Funktion ist vorhanden, jedoch verletzen die Merkmale des Operanden die Einschränkungen der Funktion. Ein Beispiel für diesen Fall ist der relationale Operator IS NULL. Die meisten Datenquellen unterstützen ihn, aber einige haben möglicherweise Einschränkungen, wie zum Beispiel, dass nur ein Spaltenname auf der linken Seite des Operators IS NULL zulässig ist.
- Die Funktion liefert möglicherweise ein anderes Ergebnis, wenn sie fern ausgewertet wird. Ein Beispiel für diesen Fall ist der Operator '>' (größer als). Für Datenquellen mit abweichenden Sortierfolgen kann der Operator 'größer als' andere Ergebnisse liefern als bei einer lokalen Auswertung durch DB2.

Kurznamenmerkmale mit Auswirkung auf die Pushdown-Möglichkeiten

Die folgenden kurznamenspezifischen Faktoren können sich auf die Pushdown-Möglichkeiten auswirken.

Lokaler Datentyp einer Kurznamenspalte: Stellen Sie sicher, dass der lokale Datentyp einer Spalte nicht verhindert, dass ein Vergleichselement an der Datenquelle ausgewertet werden kann. Verwenden Sie die Standarddatentypenzuordnungen, um einen möglichen Überlauf zu vermeiden. Jedoch wird ein verknüpfendes Vergleichselement zwischen zwei Spalten unterschiedlicher Längen eventuell nicht an der Datenquelle ausgewertet, deren Verknüpfungsspalte kürzer ist, je nachdem, wie DB2 die längere Spalte bindet. Dieser Umstand kann sich auf die Anzahl der Möglichkeiten auswirken, die das DB2-Optimierungsprogramm in einer Verknüpfungssequenz auswerten kann. Zum Beispiel erhalten Spalten einer Oracle-Datenquelle, die mit dem Datentyp INTEGER bzw. INT erstellt wurden, den Typ NUMBER(38). Eine Kurznamenspalte für diesen Oracle-Datentyp erhält den lokalen Datentyp FLOAT, weil die Werte einer ganzen Zahl in DB2 den Bereich von 2**31 bis (-2**31)-1 umfassen, was grob dem Typ NUMBER(9) entspricht. In diesem Fall können Verknüpfungen zwischen einer DB2-Spalte des Typs INTEGER und einer Oracle-Spalte des Typs INTEGER nicht an der DB2-Datenquelle stattfinden (kürzere Verknüpfungsspalte). Wenn jedoch der Wertebereich dieser Oracle-Spalte des Typs INTEGER in dem DB2-Datentyp INTEGER untergebracht werden kann, ändern Sie den lokalen Datentyp der Spalte mit der Anweisung ALTER NICKNAME, so dass die Verknüpfung an der DB2-Datenquelle stattfinden kann.

Spaltenoptionen: Verwenden Sie die SQL-Anweisung ALTER NICKNAME, um Spaltenoptionen für Kurznamen hinzuzufügen oder zu ändern.

Verwenden Sie die Option *varchar_no_trailing_blanks* zur Angabe einer Spalte, die keine folgenden Leerzeichen enthält. Der Pushdown-Analyseschritt des Compilers berücksichtigt diese Information bei der Prüfung aller Operationen für Spalten, die so markiert sind. Aufgrund dieser Angabe kann DB2 eine andere, aber äquivalente Form eines Vergleichselements generieren, das in einer fernen, an eine Datenquelle gesendete SQL-Anweisung zu verwenden ist. Ein Benutzer bemerkt vielleicht, dass an der Datenquelle ein anderes Vergleichselement ausgewertet wird, das Nettoergebnis sollte jedoch äquivalent sein.

Verwenden Sie die Option *numeric_string* zur Angabe, ob die Werte in der betreffenden Spalte immer Ziffern ohne folgende Leerzeichen sind.

In der folgenden Tabelle werden diese Optionen beschrieben.

Tabelle 28. Spaltenoptionen und zugehörige Einstellungen

Option	Gültige Einstellungen		Standard-einstellung
numeric_string	'Y'	Der Wert 'Y' (Yes) gibt an, dass diese Spalte nur Zeichenfolgen mit numerischen Daten enthält. WICHTIG: Wenn die Spalte nur numerische Zeichenfolgen mit folgenden Leerzeichen enthält, geben Sie 'Y' nicht an.	'N'
	'N'	Der Wert 'N' (No) gibt an, dass diese Spalte nicht auf Zeichenfolgen mit numerischen Daten beschränkt ist.	
	Wenn Sie die Spaltenoption numeric_string auf den Wert 'Y' setzen, teilen Sie dem Optimierungsprogramm mit, dass diese Spalte keine Leerzeichen enthält, die einen störenden Einfluss auf das Sortieren der Daten dieser Spalte haben könnten. Diese Option ist in solchen Fällen nützlich, in denen die Sortierfolge einer Datenquelle von der Sortierfolge in DB2 abweicht. Mit dieser Option markierte Spalten werden nicht von der lokalen Auswertung (der Datenquelle) aufgrund unterschiedlicher Sortierfolgen ausgeschlossen.		
varchar_no_trailing_blanks	Gibt an, ob diese Datenquelle eine VARCHAR-Vergleichssemantik für nicht mit Leerzeichen aufgefüllte Zeichenfolgen verwendet. Für Zeichenfolgen variabler Länge, die keine folgenden Leerzeichen enthalten, liefert die Vergleichssemantik für nicht mit Leerzeichen aufgefüllte Zeichenfolgen einiger DBMSs die gleichen Ergebnisse wie die Vergleichssemantik von DB2. Wenn Sie sicher sind, dass alle VARCHAR-Spalten von Tabellen und Sichten an einer Datenquelle keine folgenden Leerzeichen enthalten, können Sie in Betracht ziehen, diese Serveroption auf den Wert 'Y' für eine Datenquelle zu setzen. Diese Option wird häufig für Datenquellen von Oracle verwendet. Stellen Sie sicher, dass Sie alle Objekte berücksichtigen, die Kurznamen haben können, einschließlich Sichten.		'N'
	'Y'	Diese Datenquelle verfügt über eine Vergleichssemantik für nicht mit Leerzeichen aufgefüllte Zeichenfolgen, die der Vergleichssemantik von DB2 ähnlich ist.	
	'N'	Diese Datenquelle verfügt über keine mit DB2 vergleichbare Vergleichssemantik für nicht mit Leerzeichen aufgefüllte Zeichenfolgen.	

Abfragemerkmale mit Auswirkung auf die Pushdown-Möglichkeiten

Eine Abfrage kann einen SQL-Operator enthalten, der Kurznamen aus verschiedenen Datenquellen mit einbezieht. Die Operation muss in DB2 stattfinden, um die Ergebnisse aus zwei angegebenen Datenquellen zu kombinieren, die nur einen Operator verwenden, wie zum Beispiel einen Gruppenoperator (z. B. UNION). Der Operator kann nicht direkt an der fernen Datenquelle ausgewertet werden.

Zugehörige Konzepte:

- „Richtlinien zur Analyse, wo eine Abfrage für zusammengeschlossene Datenbanken ausgewertet wird" auf Seite 215

Richtlinien zur Analyse, wo eine Abfrage für zusammengeschlossene Datenbanken ausgewertet wird

DB2® stellt zwei Dienstprogramme bereit, die zeigen, wo Abfragen ausgewertet werden:

- Visual Explain. Starten Sie dieses Programm über den Befehl **db2cc**. Dieses Programm dient zum Anzeigen des Zugriffsplandiagramms. Die Ausführungsposition für jeden Operator ist der detaillierten Anzeige für einen Operator zu entnehmen.

  Wenn eine Abfrage an eine Datenquelle verschoben wird (Pushdown), sollten Sie einen Operator RETURN sehen. Der Operator RETURN ist ein Standardoperator von DB2. Für eine SELECT-Anweisung, die Daten aus einem Kurznamen auswählt, sehen Sie außerdem einen Operation SHIP. Der Operator SHIP ist für Operationen mit zusammengeschlossenen Datenbanken spezifisch. Er ändert das Servermerkmal des Datenflusses und trennt lokale Operatoren von fernen Operatoren. Die SELECT-Anweisung wird mit Hilfe der von der Datenquelle unterstützten SQL-Version generiert. Sie kann jede für die Datenquelle gültige Abfrage enthalten.

  Wenn eine INSERT-, DELETE- oder UPDATE-Abfrage vollständig an die ferne Datenbank verschoben werden kann (Pushdown), wird eventuell keine SHIP-Anweisung im Zugriffsplan ausgewiesen. Alle fern ausgeführten INSERT-, DELETE- oder UPDATE-Anweisungen werden jedoch für den Operator RETURN gezeigt. Wenn eine Abfrage jedoch nicht vollständig verschoben werden kann, zeigt der Operator SHIP, welche Operationen fern durchgeführt wurden.
- SQL-EXPLAIN. Starten Sie dieses Programm mit dem Befehl **db2expln** oder **dynexpln**. Dieses Programm dient zur Erstellung einer Textausgabe des Zugriffsplans.

Verstehen, warum eine Abfrage an einer Datenquelle oder in DB2 ausgewertet wird

Berücksichtigen Sie die folgenden Schlüsselfragen, wenn Sie Methoden zur Ausweitung der Pushdown-Möglichkeiten untersuchen:

- Warum wird dieses Vergleichselement nicht fern ausgewertet?

  Diese Frage erhebt sich, wenn ein Vergleichselement sehr selektiv ist und daher zum Filtern von Zeilen herangezogen werden und den Netzwerkverkehr verringern könnte. Die ferne Auswertung von Vergleichselementen wirkt sich auch darauf aus, ob eine Verknüpfung zwischen zwei Tabellen derselben Datenquelle fern ausgewertet werden kann.

  Zu untersuchende Bereiche sind:

  – Vergleichselemente mit Unterabfragen. Enthält dieses Vergleichselement eine Unterabfrage, die sich auf eine andere Datenquelle bezieht? Enthält dieses Vergleichselement eine Unterabfrage mit einem SQL-Operator, der von dieser Datenquelle nicht unterstützt wird? Nicht alle Datenquellen unterstützen Gruppenoperatoren in einem Vergleichselement einer Unterabfrage.
  – Funktionen in Vergleichselementen. Enthält dieses Vergleichselement eine Funktion, die von dieser fernen Datenquelle nicht ausgewertet werden kann? Relationale Operatoren werden als Funktionen klassifiziert.
  – Bindeanforderungen von Vergleichselementen. Erfordert dieses Vergleichselement, wenn es fern ausgewertet werden soll, das Einbinden eines bestimmten Werts? Ist dies der Fall, würde der Wert die SQL-Einschränkungen an dieser Datenquelle verletzen?

- Globale Optimierung. Das Optimierungsprogramm kann entschieden haben, dass der Aufwand bei lokaler Verarbeitung geringer ist.

- Warum wird der Operator GROUP BY nicht fern ausgewertet?

  Es gibt verschiedene Bereiche, die überprüft werden können:

  - Wird die Eingabe für den Operator GROUP BY fern ausgewertet? Ist dies zu verneinen, untersuchen Sie die Eingabe.
  - Besitzt die Datenquelle Einschränkungen für diesen Operator? Beispiele hierfür sind:
    - Begrenzte Anzahl von GROUP BY-Elementen
    - Begrenzte Byteanzahl kombinierter GROUP BY-Elemente
    - Spaltenspezifikation nur für die GROUP BY-Liste
  - Unterstützt die Datenquelle diesen SQL-Operator?
  - Globale Optimierung. Das Optimierungsprogramm kann entschieden haben, dass der Aufwand bei lokaler Verarbeitung geringer ist.
  - Enthält die Operatorklausel GROUP BY einen Zeichenausdruck? Ist dies der Fall, überprüfen Sie, ob die ferne Datenquelle die gleichen Einstellungen für die Beachtung der Groß-/Kleinschreibung wie DB2 verwendet.

- Warum wird der Gruppenoperator nicht fern ausgewertet?

  Es gibt verschiedene Bereiche, die überprüft werden können:

  - Werden beide Operanden vollständig an derselben Datenquelle ausgewertet? Ist dies nicht der Fall, obwohl es der Fall sein sollte, untersuchen Sie jeden Operanden.
  - Besitzt die Datenquelle Einschränkungen für diesen Gruppenoperator? Sind zum Beispiel große Objekte oder Langfelder gültige Eingaben für diesen speziellen Gruppenoperator?

- Warum wird die Operation ORDER BY nicht fern ausgeführt?

  Betrachten Sie Folgendes:

  - Wird die Eingabe für die Operation ORDER BY fern ausgewertet? Ist dies zu verneinen, untersuchen Sie die Eingabe.
  - Enthält die Klausel ORDER BY einen Zeichenausdruck? Wenn ja, besitzt die ferne Datenquelle nicht die gleiche Sortierfolge oder Einstellung zur Beachtung der Groß-/Kleinschreibung wie DB2?
  - Besitzt die Datenquelle Einschränkungen für diesen Operator? Gibt es beispielsweise eine begrenzte Anzahl von ORDER BY-Elementen? Schränkt die Datenquelle die Spaltenangaben für die ORDER BY-Liste ein?

Zugehörige Konzepte:

- „Richtlinien zur Zeichenkonvertierung" auf Seite 101
- „Globale Analyse von Abfragen auf zusammengeschlossene Datenbanken" auf Seite 220
- „Generierung von fernem SQL und globale Optimierung in zusammengeschlossenen Datenbanken" auf Seite 217
- „Informationen zu Abfragen auf zusammengeschlossene Datenbanken" auf Seite 662

Generierung von fernem SQL und globale Optimierung in zusammengeschlossenen Datenbanken

Für eine Abfrage auf eine zusammengeschlossenen Datenbank, die relationale Kurznamen verwendet, kann die Zugriffsstrategie vorsehen, die Originalabfrage in eine Gruppe ferner Abfrageeinheiten zu zerlegen und anschließend die Ergebnisse zu kombinieren. Diese Generierung von fernem SQL hilft bei der Herstellung einer global optimalen Zugriffsstrategie für eine Abfrage.

Das Optimierungsprogramm verwendet die Ausgabe der Pushdown-Analyse zur Entscheidung, ob die jeweilige Operation lokal in DB2® oder fern an einer Datenquelle ausgewertet wird. Es stützt die Entscheidung auf die Ausgabe des Aufwandsmodells, das nicht nur den Aufwand für die Auswertung der Operation, sondern auch den Aufwand für die Übertragung der Daten oder Nachrichten zwischen DB2 und den Datenquellen berücksichtigt.

Obwohl das Ziel die Erstellung einer optimierten Abfrage ist, wird die Ausgabe aus der globalen Optimierung und somit auch die Leistung der Abfrage von den folgenden Hauptfaktoren beeinflusst.

- Servermerkmale
- Kurznamenmerkmale

Servermerkmale und Optionen mit Auswirkung auf die globale Optimierung

Die folgenden Serverfaktoren einer Datenquelle können sich auf die globale Optimierung auswirken:

- Relatives Verhältnis der CPU-Geschwindigkeiten

  Geben Sie mit Hilfe der Serveroption *cpu_ratio* an, um wie schnell oder langsam die CPU-Geschwindigkeit der Datenquelle im Verhältnis zur CPU von DB2 ist. Ein niedriger Wert bedeutet, dass die CPU des Computers der Datenquelle schneller als die CPU des DB2-Computers ist. Wenn der Verhältniswert niedrig ist, zieht das DB2-Optimierungsprogramm mit größerer Wahrscheinlichkeit in Betracht, CPU-intensive Operationen an die Datenquelle zu verschieben (Pushdown).
- Relatives Verhältnis der E/A-Geschwindigkeiten

  Geben Sie mit Hilfe der Serveroption *io_ratio* an, um wie viel schneller oder langsamer die E/A-Geschwindigkeit des Datenquellensystems im Verhältnis zum DB2-System ist. Ein niedriger Wert bedeutet, dass die E/A-Geschwindigkeit der Workstation der Datenquelle schneller als die E/A-Geschwindigkeit der DB2-Workstation ist. Wenn der Verhältniswert niedrig ist, zieht das DB2-Optimierungsprogramm in Betracht, E/A-intensive Operationen an die Datenquelle zu verschieben (Pushdown).
- Übertragungsgeschwindigkeit zwischen DB2 und der Datenquelle

  Geben Sie mit Hilfe der Serveroption *comm_rate* die Netzwerkkapazität an. Langsame Geschwindigkeiten, die eine langsame Kommunikation zwischen DB2 und der Datenquelle bedeuten, veranlassen das DB2-Optimierungsprogramm, die Anzahl der Nachrichten zu reduzieren, die zwischen DB2 und dieser Datenquelle gesendet werden. Wenn die Geschwindigkeit auf den Wert 0 gesetzt wird, erstellt das Optimierungsprogramm einen Zugriffsplan, der nur minimalen Netzwerkverkehr erfordert.

- Sortierfolge der Datenquelle

  Geben Sie mit Hilfe der Serveroption *collating_sequence* an, ob die Sortierfolge einer Datenquelle mit der lokalen DB2-Sortierfolge übereinstimmt. Wenn die Option nicht auf den Wert 'Y' gesetzt ist, betrachtet das Optimierungsprogramm die von dieser Datenquelle abgerufenen Daten als unsortiert.
- Ferne Planhinweise

  Geben Sie mit Hilfe der Serveroption *plan_hints* an, dass Planhinweise generiert werden oder an einer Datenquelle verwendet werden sollen. Standardmäßig sendet DB2 keine Planhinweise an die Datenquelle.

  Planhinweise sind Anweisungsfragmente, die zusätzliche Informationen für Optimierungsprogramme von Datenquellen bereitstellen. Für bestimmte Abfragen können diese Informationen die Leistung verbessern. Die Planhinweise können das Optimierungsprogramm der Datenquelle bei der Entscheidung unterstützen, ob ein Index, und wenn ja, welcher, zu verwenden ist oder nach welcher Reihenfolge bei der Verknüpfung von Tabellen vorzugehen ist.

  Wenn Planhinweise aktiviert sind, enthält die Abfrage, die an die Datenquelle gesendet wird, zusätzliche Informationen. Zum Beispiel könnte eine Anweisung, die an ein Oracle-Optimierungsprogramm mit Planhinweisen gesendet wird, folgendermaßen aussehen:

  ```
  SELECT /*+ INDEX (tabelle1, t1index)*/
     col1
     FROM tabelle1
  ```

  Der Planhinweis ist hier die Zeichenfolge `/*+ INDEX (tabelle1, t1index)*/`.
- Informationen in der Wissensbasis des DB2-Optimierungsprogramms

  DB2 besitzt eine Wissensbasis (engl. knowledge base) für das Optimierungsprogramm, die Daten über spezifische Datenquellen enthält. Das DB2-Optimierungsprogramm generiert keine fernen Zugriffspläne, die nicht von spezifischen Datenbankverwaltungssystemen (DBMSs) generiert werden können. Mit anderen Worten, DB2 vermeidet die Generierung von Plänen, die von Optimierungsprogrammen an fernen Datenquellen nicht verstanden bzw. akzeptiert werden.

Kurznamenmerkmale mit Auswirkung auf die globale Optimierung

Die folgenden kurznamenspezifischen Faktoren können sich auf die globale Optimierung auswirken.

Indexinformationen: DB2 kann Informationen über Indizes an Datenquellen zur Optimierung von Abfragen heranziehen. Daher ist es wichtig, dass die für DB2 verfügbaren Indexinformationen aktuell sind. Die Indexinformationen für Kurznamen werden zu Anfang bei der Erstellung des Kurznamens erfasst. Indexinformationen werden für Sichtkurznamen nicht gesammelt.

Erstellen von Indexspezifikationen für Kurznamen: Sie können eine Indexspezifikation für einen Kurznamen erstellen. Indexspezifikationen bilden eine Indexdefinition (keinen tatsächlichen Index) im Katalog, die vom DB2-Optimierungsprogramm verwendet werden kann. Indexspezifikationen werden mit der Anweisung CREATE INDEX SPECIFICATION ONLY erstellt. Die Syntax für die Erstellung einer Indexspezifikation für einen Kurznamen ist der Syntax zur Erstellung eines Index für eine lokale Tabelle ähnlich.

Ziehen Sie die Erstellung von Indexspezifikationen in folgenden Fällen in Betracht:
- DB2 kann während der Kurznamenerstellung keine Indexinformationen von einer Datenquelle abrufen.
- Sie wünschen einen Index für einen Sichtkurznamen.
- Sie wollen das DB2-Optimierungsprogramm veranlassen, einen speziellen Kurznamen als innere Tabelle einer Verknüpfung mit Verschachtelungsschleife zu verwenden. Der Benutzer kann einen Index für die Verknüpfungsspalte erstellen, falls keiner vorhanden ist.

Überlegen Sie vor dem Ausführen von Anweisungen CREATE INDEX für einen Kurznamen einer Sicht, ob Sie den Index benötigen. Wenn die Sicht eine einfache SELECT-Abfrage auf eine Tabelle mit einem Index ist, kann die Erstellung lokaler Indizes für den Kurznamen, die mit den Indizes für die Tabelle an der Datenquelle übereinstimmen, die Abfrageleistung erheblich erhöhen. Wenn Indizes jedoch lokal für Sichten erstellt werden, die keine einfachen SELECT-Anweisungen enthalten, zum Beispiel eine Sicht, die durch Verknüpfen zweier Tabellen erstellt wird, kann die Abfrageleistung sinken. Wenn Sie zum Beispiel einen Index für eine Sicht erstellen, die auf der Verknüpfung zweier Tabellen basiert, kann das Optimierungsprogramm diese Sicht möglicherweise als inneres Element in einer Verknüpfung über Verschachtelungsschleife (Nested Loop Join) wählen. Die Abfrage wird eine geringe Leistung erzielen, weil in diesem Fall die Verknüpfung mehrere Male ausgewertet wird. Eine Alternative wäre, Kurznamen für jede der Tabellen, auf die in der Sicht der Datenquelle verwiesen wird, zu erstellen und eine lokale Sicht unter DB2 zu definieren, die auf beide Kurznamen verweist.

Katalogstatistiken: Katalogstatistiken beschreiben die allgemeine Größe von Kurznamen und den Wertebereich in den zugehörigen Spalten. Das Optimierungsprogramm verwendet diese Statistikdaten, wenn es den Pfad mit dem geringsten Aufwand zur Verarbeitung von Abfragen berechnet, die Kurznamen enthalten. Die statistischen Daten über Kurznamen werden in den gleichen Katalogsichten wie Tabellenstatistiken gespeichert.

Obwohl DB2 die an einer Datenquelle gespeicherten statistischen Daten abrufen kann, kann DB2 Aktualisierungen an vorhandenen Statistikdaten an Datenquellen nicht automatisch erkennen. Überdies kann DB2 Änderungen in Objektdefinitionen oder strukturelle Änderungen wie das Hinzufügen einer Spalte an Objekten von Datenquellen nicht verarbeiten. Wenn die Statistik- bzw. Strukturdaten für ein Objekt geändert wurden, haben Sie zwei Möglichkeiten:
- Führen Sie an der Datenquelle das Programm aus, das dem Dienstprogramm RUNSTATS entspricht. Löschen Sie anschließend den aktuellen Kurznamen und erstellen Sie ihn neu. Verwenden Sie diese Methode, wenn sich Strukturinformationen geändert haben.
- Aktualisieren Sie die Statistiken in der Sicht SYSSTAT.TABLES manuell. Diese Methode erfordert zwar weniger Schritte, funktioniert jedoch nicht, wenn sich Strukturinformationen geändert haben.

Zugehörige Konzepte:
- „Serveroptionen mit Einfluss auf zusammengeschlossene Datenbanken" auf Seite 111
- „Richtlinien zur Analyse, wo eine Abfrage für zusammengeschlossene Datenbanken ausgewertet wird" auf Seite 215
- „Globale Analyse von Abfragen auf zusammengeschlossene Datenbanken" auf Seite 220

Globale Analyse von Abfragen auf zusammengeschlossene Datenbanken

Die beiden folgenden Dienstprogramme sind zum Anzeigen globaler Zugriffspläne verfügbar:

- Visual Explain. Starten Sie dieses Programm über die Steuerzentrale oder führen Sie den Befehl *db2cc* aus, der die Steuerzentrale startet. Verwenden Sie Visual Explain zum Anzeigen des Zugriffsplandiagramms für eine Abfrage. Die Ausführungsposition für jeden Operator ist der detaillierten Anzeige für einen Operator zu entnehmen. Sie können darüber hinaus die ferne SQL-Anweisung, die für die jeweilige Datenquelle generiert wurde, je nach Art der Abfrage dem Operator SHIP oder RETURN entnehmen. Durch eine Untersuchung der Details für jeden Operator können Sie die Anzahl der Zeilen feststellen, die vom DB2®-Optimierungsprogramm als Eingabe für den jeweiligen Operator und als Ausgabe aus diesem Operator geschätzt wird. Weiterhin können Sie den für die Ausführung des jeweiligen Operators geschätzten Aufwand, einschließlich des Kommunikationsaufwands, ablesen.
- SQL-EXPLAIN. Starten Sie das Programm mit dem Befehl *db2expln* oder *dynexpln*. SQL-EXPLAIN dient zur Erstellung einer Textausgabe des Zugriffsplans. Obwohl SQL-EXPLAIN keine Informationen über den Aufwand bietet, können Sie der Ausgabe den Zugriffsplan entnehmen, der von dem fernen Optimierungsprogramm für die Datenquellen generiert wurde, die von der fernen EXPLAIN-Funktion unterstützt werden.

Verstehen der Entscheidungen der DB2-Optimierung

Berücksichtigen Sie bei der Untersuchung von Leistungsverbesserungen die folgenden Fragen und Schlüsselbereiche der Optimierung:

- Warum wird eine Verknüpfung zwischen zwei Kurznamen derselben Datenquelle nicht fern an der Datenquelle ausgewertet?

  Zu untersuchende Bereiche sind:
  - Verknüpfungsoperationen. Werden sie von der Datenquelle unterstützt?
  - Verknüpfungsvergleichselemente. Können die Verknüpfungsvergleichselemente an der fernen Datenquelle ausgewertet werden? Wenn dies zu verneinen ist, untersuchen Sie das Verknüpfungsvergleichselement.
  - Anzahl der Zeilen im Verknüpfungsergebnis (mit Visual Explain). Ergibt sich aus der Verknüpfung eine wesentlich größere Gruppe von Zeilen als durch die Kombination der beiden Kurznamen? Ist die jeweilige Anzahl sinnvoll? Wenn dies zu verneinen ist, ziehen Sie eine manuelle Aktualisierung der Kurznamenstatistik (SYSSTAT.TABLES) in Betracht.
- Warum wird der Operator GROUP BY nicht fern ausgewertet?

  Zu untersuchende Bereiche sind:
  - Operatorsyntax. Stellen Sie sicher, dass der Operator an der fernen Datenquelle ausgewertet werden kann.
  - Anzahl von Zeilen. Prüfen Sie die geschätzte Anzahl von Zeilen in der Eingabe und Ausgabe des Operators GROUP BY mit Hilfe von Visual Explain. Liegt die eine Anzahl sehr nahe an der anderen? Ist dies zu bejahen, betrachtet das DB2-Optimierungsprogramm es als effizienter, diesen Operator GROUP BY lokal auszuwerten. Ist die jeweilige Anzahl zudem sinnvoll? Wenn dies zu verneinen ist, ziehen Sie eine manuelle Aktualisierung der Kurznamenstatistik (SYSSTAT.TABLES) in Betracht.

- Warum wird die Anweisung nicht vollständig durch die ferne Datenquelle ausgewertet?

  Das DB2-Optimierungsprogramm führt eine aufwandsorientierte Optimierung durch. Auch wenn die Pushdown-Analyse ergibt, dass jeder Operator an der fernen Datenquelle ausgewertet werden kann, stützt sich das Optimierungsprogramm bei der Generierung eines global optimalen Plans auf die Aufwandsschätzung. Es gibt eine Vielzahl von Faktoren, die in diesen Plan einfließen können. Es ist beispielsweise möglich, dass eine ferne Datenquelle zwar jede einzelne Operation in der ursprünglichen Abfrage ausführen kann, aber die CPU-Geschwindigkeit der Datenquelle wesentlich langsamer ist als die CPU-Geschwindigkeit des DB2-Systems, so dass es vorteilhafter ist, die Operationen in DB2 lokal auszuführen. Wenn die Ergebnisse nicht zufrieden stellend sind, überprüfen Sie die Serverstatistikdaten in SYSCAT.SERVEROPTIONS.
- Warum zeigt ein vom Optimierungsprogramm generierter Plan, der vollständig an einer fernen Datenquelle ausgewertet wird, eine wesentlich schlechtere Leistung als die Originalabfrage bei direkter Ausführung an der fernen Datenquelle?

  Zu untersuchende Bereiche sind:

  – Die vom DB2-Optimierungsprogramm generierte ferne SQL-Anweisung. Überprüfen Sie, ob sie mit der Originalabfrage identisch ist. Suchen Sie nach Änderungen in der Reihenfolge der Vergleichselemente. Ein gutes Abfrageoptimierungsprogramm sollte nicht von der Reihenfolge der Vergleichselemente einer Abfrage abhängig sein. Leider sind nicht alle DBMS-Optimierungsprogramme identisch, so dass das Optimierungsprogramm der fernen Datenquelle vielleicht aufgrund der Reihenfolge der Vergleichselemente in der Eingabe einen anderen Plan generiert. Wenn dies der Fall ist, liegt das Problem bei dem fernen Optimierungsprogramm. Sie können in diesem Fall entweder eine Änderung der Reihenfolge der Vergleichselemente in der Eingabe für DB2 in Betracht ziehen oder die Serviceorganisation der fernen Datenquelle um Hilfe bitten.

    Prüfen Sie darüber hinaus auf Ersetzungen von Vergleichselementen. Ein gutes Abfrageoptimierungsprogramm sollte nicht von der Ersetzung äquivalenter Vergleichselemente abhängig sein. Leider sind nicht alle DBMS-Optimierungsprogramme identisch, so dass das Optimierungsprogramm der fernen Datenquelle vielleicht aufgrund des Vergleichselements in der Eingabe einen anderen Plan generiert. Zum Beispiel können einige Optimierungsprogramme keine Anweisungen für Vergleichselemente unter Verwendung der Transitivität generieren.

  – Die Anzahl der zurückgegebenen Zeilen. Diese Anzahl kann mit Hilfe von Visual Explain festgestellt werden. Wenn die Abfrage eine große Anzahl von Zeilen zurückgibt, kann der Netzwerkverkehr zum potenziellen Engpass werden.

  – Zusätzliche Funktionen. Enthält die ferne SQL-Anweisung im Vergleich zur Originalabfrage zusätzliche Funktionen? Es können einige zusätzliche Funktionen zur Umsetzung von Datentypen generiert werden. Stellen Sie sicher, dass diese erforderlich sind.

Zugehörige Konzepte:

- „Serveroptionen mit Einfluss auf zusammengeschlossene Datenbanken" auf Seite 111
- „Pushdown-Analyse für zusammengeschlossene Datenbanken" auf Seite 209
- „Richtlinien zur Analyse, wo eine Abfrage für zusammengeschlossene Datenbanken ausgewertet wird" auf Seite 215
- „Generierung von fernem SQL und globale Optimierung in zusammengeschlossenen Datenbanken" auf Seite 217
- „Informationen zu Abfragen auf zusammengeschlossene Datenbanken" auf Seite 662
- „Beispiel 5: Zugriffsplan für eine zusammengeschlossene Datenbank" auf Seite 674

Kapitel 7. Die SQL-EXPLAIN-Einrichtung

Die EXPLAIN-Einrichtung gibt Ihnen die Möglichkeit, Informationen über den vom Optimierungsprogramm gewählten Zugriffsplan sowie Leistungsinformationen zu Zwecken der Abfrageoptimierung zu erfassen.

SQL-EXPLAIN-Einrichtung

Der SQL-Compiler kann Informationen über den Zugriffsplan und die Umgebung statischer oder dynamischer SQL-Anweisungen erfassen. Die erfassten Informationen helfen Ihnen zu verstehen, wie einzelne SQL-Anweisungen ausgeführt werden, so dass Sie die Anweisungen und die Konfiguration des Datenbankmanagers zur Leistungsverbesserung optimieren können.

Sie können EXPLAIN-Daten zu folgenden Zwecken erfassen und verwenden:

- Um zu verstehen, wie der Datenbankmanager auf Tabellen und Indizes zur Erfüllung Ihrer Abfrage zugreift.
- Um Ihre Maßnahmen zur Leistungsverbesserung zu beurteilen.

  Wenn Sie einen Aspekt des Datenbankmanagers, der SQL-Anweisungen oder der Datenbank ändern, sollten Sie die EXPLAIN-Daten untersuchen, um herauszufinden, wie sich Ihre Maßnahme auf die Leistung ausgewirkt hat.

Zu den erfassten Informationen gehören folgende:

- Informationen über die Reihenfolge der Operationen zur Verarbeitung der Abfrage
- Informationen über den Aufwand
- Vergleichselemente und Selektivitätsschätzwerte für jedes Vergleichselement
- Statistiken für alle Objekte, auf die in der SQL-Anweisung zum Zeitpunkt der EXPLAIN-Datenerfassung verwiesen wird
- Werte für die Hostvariablen, Parametermarken oder Sonderregister, die zur Reoptimierung der SQL-Anweisung verwendet werden.

Bevor Sie EXPLAIN-Informationen erfassen können, erstellen Sie die relationalen Tabellen, in denen das Optimierungsprogramm die EXPLAIN-Informationen speichert, und definieren die Sonderregister, die die Art der zu erfassenden EXPLAIN-Informationen festlegen.

Zum Anzeigen von EXPLAIN-Informationen können Sie entweder ein Befehlszeilentool oder Visual Explain verwenden. Definieren Sie die Registriervariablen, die festlegen, welche EXPLAIN-Daten erfasst werden, entsprechend dem Tool, das Sie verwenden. Wenn Sie zum Beispiel davon ausgehen, dass Sie nur Visual Explain verwenden, brauchen Sie nur Momentaufnahmeninformationen zu erfassen. Wenn Sie beabsichtigen, eine detaillierte Analyse mit einem der Befehlszeilendienstprogramme oder mit angepassten SQL-Anweisungen für die EXPLAIN-Tabellen durchzuführen, sollten Sie alle EXPLAIN-Informationen erfassen.

Zugehörige Konzepte:

- „EXPLAIN-Tools" auf Seite 224
- „Richtlinien zur Verwendung von EXPLAIN-Informationen" auf Seite 226

- „Die EXPLAIN-Tabellen und die Organisation von EXPLAIN-Informationen" auf Seite 227
- „Richtlinien zur Erfassung von EXPLAIN-Informationen" auf Seite 234
- „Richtlinien zur Analyse von EXPLAIN-Informationen" auf Seite 236
- „SQL-EXPLAIN-Tools" auf Seite 637

Tools zur Erfassung und Analyse von EXPLAIN-Informationen

Die EXPLAIN-Einrichtung kann auf verschiedene Weisen ausgeführt werden, je nachdem, welche Art von Leistungsanalyse durchgeführt werden soll. Dieser Abschnitt beschreibt die Tools, die die EXPLAIN-Einrichtung implementieren, sowie die Möglichkeiten, wie Sie die erfassten Informationen nutzen können.

EXPLAIN-Tools

DB2® stellt eine umfassende EXPLAIN-Einrichtung bereit, die detaillierte Informationen über den Zugriffsplan liefert, den das Optimierungsprogramm für eine SQL-Anweisung auswählt. Der Zugriff auf die Tabellen, in denen EXPLAIN-Daten, d. h. Informationen zu statischen und dynamischen SQL-Anweisungen, gespeichert werden, ist auf allen unterstützten Plattformen möglich. Verschiedene Tools oder Methoden bieten flexible Möglichkeiten, die EXPLAIN-Informationen zu erfassen, anzuzeigen und zu analysieren.

Detaillierte Informationen des Optimierungsprogramms, die eine eingehende Analyse eines Zugriffsplans ermöglichen, werden getrennt vom eigentlichen Zugriffsplan in EXPLAIN-Tabellen gespeichert. Es stehen mehrere Methoden zum Abrufen von Informationen aus den EXPLAIN-Tabellen zur Verfügung:

- Visual Explain zum Anzeigen von EXPLAIN-Momentaufnahmen

  Rufen Sie Visual Explain über die Steuerzentrale auf, um eine grafische Darstellung eines Abfragezugriffsplans anzuzeigen. Sie können sowohl statische als auch dynamische SQL-Anweisungen analysieren.

  Visual Explain ermöglicht Ihnen, auf einer anderen Plattform erfasste oder erstellte Momentaufnahmen anzuzeigen. Zum Beispiel kann ein Windows® NT-Client Momentaufnahmen darstellen, die auf einem Server unter DB2 für HP-UX generiert wurden. Dazu müssen beide Plattformen jedoch die Version 5 oder höher haben.
- Das Tool *db2exfmt* zum Anzeigen von EXPLAIN-Informationen in einer vorformatierten Ausgabe
- Die Tools *db2expln* und *dynexpln*

  Zugriffsplaninformationen, die für ein oder mehrere Pakete mit statischen SQL-Anweisungen verfügbar sind, können mit dem Programm *db2expln* über die Befehlszeile angezeigt werden. Das Tool *db2expln* zeigt die tatsächliche Implementierung des gewählten Zugriffsplans. Informationen des Optimierungsprogramms werden nicht angezeigt.

  Das Programm *dynexpln*, das intern auf *db2expln* zurückgreift, stellt eine schnelle Methode zur Analyse dynamischer SQL-Anweisungen dar, die keine Parametermarken enthalten. Die Verwendung von *db2expln* innerhalb des Programms *dynexpln* wird durch die Transformation der Eingabeanweisung in eine statische SQL-Anweisung innerhalb eines Pseudopakets ermöglicht. Wenn dieses Verfahren angewandt wird, sind die Informationen nicht immer ganz exakt. Wenn es auf Genauigkeit ankommt, verwenden Sie die EXPLAIN-Einrichtung.

Das Tool *db2expln* liefert einen relativ kompakten und englisch aufbereiteten Überblick über die Operationen, die bei der Ausführung stattfinden, indem der tatsächlich generierte Zugriffsplan analysiert wird.

- Schreiben eigener Abfragen für die EXPLAIN-Tabellen

  Durch Schreiben eigener Abfragen können Sie die Ausgabe leicht bearbeiten und verschiedene Abfragen oder auch die gleiche Abfrage über einen Zeitraum hinweg vergleichen.

Anmerkung: Die EXPLAIN-Tools und andere Tools, die über die Befehlszeile ausgeführt werden, wie *db2batch*, *dynexpln*, *db2vexp* und *db2_all*, befinden sich im Unterverzeichnis *misc* des Verzeichnisses *sqllib*. Wenn die Tools aus diesem Pfad an eine andere Position versetzt werden, funktionieren die Befehlszeilenmethoden möglicherweise nicht.

Die folgende Tabelle enthält eine Übersicht über die verschiedenen, in der EXPLAIN-Einrichtung von DB2 verfügbaren Tools und ihre Merkmale. Verwenden Sie die Tabelle zur Auswahl des Tools, das für Ihre Anforderungen und Ihre Umgebung am besten geeignet ist.

Tabelle 29. Tools der EXPLAIN-Einrichtung

Merkmale	Visual Explain	EXPLAIN-Tabellen	db2exfmt	db2expln	dynexpln
Grafische Benutzerschnittstelle	Ja				
Textausgabe			Ja	Ja	Ja
Schnelle Grobanalyse für statisches SQL				Ja	
Unterstützung für statisches SQL	Ja	Ja	Ja	Ja	
Unterstützung für dynamisches SQL	Ja	Ja	Ja	Ja	Ja*
Unterstützung für CLI-Anwendungen	Ja	Ja	Ja		
Verfügbar für DRDA®-Anwendungsrequester		Ja			
Detaillierte Informationen des Optimierungsprogramms	Ja	Ja	Ja		
Geeignet zur Analyse mehrerer Anweisungen		Ja	Ja	Ja	Ja
Zugriff auf die Informationen aus einer Anwendung heraus		Ja			

Anmerkung:

* Verwendet indirekt db2expln; es gelten einige Einschränkungen.

Zugehörige Konzepte:

- „dynexpln" auf Seite 645
- „Beschreibung der Ausgabe von db2expln und dynexpln" auf Seite 645
- „Beispiele für die Ausgabe von db2expln und dynexpln" auf Seite 665

Zugehörige Referenzen:

- Anhang D, „db2exfmt - Formatierungstool für EXPLAIN-Tabellen", auf Seite 677
- „db2expln - SQL-EXPLAIN-Tool" auf Seite 638

Richtlinien zur Verwendung von EXPLAIN-Informationen

EXPLAIN-Informationen dienen in erster Linie den beiden folgenden Zwecken:

- Analysieren der Gründe, aus denen sich die Anwendungsleistung geändert hat
- Beurteilen von Maßnahmen zur Leistungsoptimierung

Analysieren von Leistungsänderungen

Um sich ein besseres Verständnis für die Ursachen für Änderungen in der Abfrageleistung zu verschaffen, benötigen Sie EXPLAIN-Informationen vor und nach den Änderungen, die Sie durch die folgenden Schritte erhalten können:

- Erfassen Sie EXPLAIN-Informationen für die Abfrage, bevor Sie Änderungen vornehmen, und sichern Sie die resultierenden EXPLAIN-Tabellen. Alternativ könnten Sie auch die Ausgabe aus dem EXPLAIN-Tool `db2exfmt` sichern.
- Sichern oder drucken Sie die aktuellen Katalogstatistiken, wenn Sie zum Anzeigen der entsprechenden Informationen nicht auf Visual Explain zugreifen wollen oder können. Zu diesem Zweck können Sie auch das Produktivitätstool `db2look` verwenden.
- Sichern oder drucken Sie die Anweisungen der Datendefinitionssprache (DDL), einschließlich der Anweisungen für CREATE TABLE, CREATE VIEW, CREATE INDEX, CREATE TABLESPACE.

Die Informationen, die Sie auf diese Weise erfassen, stellen einen Referenzpunkt für zukünftige Analysen dar. Bei dynamischen SQL-Anweisungen können Sie diese Informationen bei der ersten Ausführung Ihrer Anwendung erfassen. Bei statischen SQL-Anweisungen können Sie diese Informationen auch beim Binden erfassen. Zur Analyse einer Leistungsänderung vergleichen Sie die Informationen, die Sie erfasst haben, mit Informationen, die Sie über die Abfrage und die Umgebung erfassen, wenn Sie Ihre Analyse beginnen.

Ein einfaches Beispiel wäre eine Analyse, die ergäbe, dass ein Index nicht mehr als Bestandteil des Zugriffspfads verwendet wird. Mit Hilfe der in Visual Explain angezeigten Informationen der Katalogstatistiken könnten Sie feststellen, dass die Anzahl von Indexstufen (Spalte NLEVELS) nun wesentlich höher ist als zu dem Zeitpunkt, als die Abfrage zum ersten Mal an die Datenbank gebunden wurde. Sie könnten in diesem Fall eine der folgenden Maßnahmen durchführen:

- Reorganisieren des Index
- Erfassen neuer Statistikdaten für die Tabelle und Indizes
- Sammeln von EXPLAIN-Informationen beim erneuten Binden der Abfrage

Nach der Durchführung einer dieser Maßnahmen untersuchen Sie den Zugriffsplan erneut. Wenn der Index wieder verwendet wird, ist die Leistung der Abfrage vielleicht kein Problem mehr. Wenn der Index weiterhin nicht verwendet wird oder die Leistung problematisch bleibt, führen Sie eine zweite Maßnahme durch und untersuchen die Ergebnisse. Wiederholen Sie diese Schritte, bis das Problem gelöst ist.

Beurteilen von Maßnahmen zur Leistungsoptimierung

Sie haben die Möglichkeit, die Abfrageleistung durch eine Reihe von Maßnahmen zu verbessern, zu denen das Anpassen von Konfigurationsparametern, das Hinzufügen von Behältern, das Erfassen aktueller Katalogstatistiken u. a. gehören.

Nach einer Änderung in einem dieser Bereiche können Sie mit der SQL-EXPLAIN-Einrichtung die Auswirkungen ermitteln, die die Änderung auf den ausgewählten Zugriffsplan hat. Wenn sie zum Beispiel einen Index oder eine gespeicherte Abfragetabelle (MQT) gemäß den Richtlinien für Indizes hinzufügen, können Sie mit Hilfe der EXPLAIN-Daten feststellen, ob der Index oder die gespeicherte Abfragetabelle tatsächlich in der erwarteten Weise genutzt wird.

Obwohl die EXPLAIN-Ausgabe Informationen liefert, die Ihnen ermöglichen, den ausgewählten Zugriffsplan und seinen relativen Aufwand zu ermitteln, besteht die einzige Möglichkeit, die Verbesserung der Leistung für eine Abfrage genau zu messen, in der Anwendung von Vergleichstesttechniken (Benchmark Tests).

Zugehörige Konzepte:

- „SQL-EXPLAIN-Einrichtung" auf Seite 223
- „Die EXPLAIN-Tabellen und die Organisation von EXPLAIN-Informationen" auf Seite 227
- „Richtlinien zur Erfassung von EXPLAIN-Informationen" auf Seite 234
- „Der Designadvisor" auf Seite 237
- „Gespeicherte Abfragetabellen" auf Seite 207

Zugehörige Referenzen:

- Anhang D, „db2exfmt - Formatierungstool für EXPLAIN-Tabellen", auf Seite 677

Erfasste EXPLAIN-Informationen

Dieser Abschnitt enthält eine Liste und Beschreibungen der einzelnen Tabellen, in denen EXPLAIN-Daten gespeichert werden, beschreibt die Art der Informationen, die Sie aus den erfassten Daten abrufen können, und erläutert einige Richtlinien zur Erfassung von Informationen, die zur Leistungsanalyse nützlich sind.

Die EXPLAIN-Tabellen und die Organisation von EXPLAIN-Informationen

Sämtliche EXPLAIN-Informationen sind nach dem Konzept eines EXPLAIN-Exemplars organisiert. Ein EXPLAIN-Exemplar stellt einen Aufruf der EXPLAIN-Einrichtung für eine oder mehrere SQL-Anweisungen dar. Die EXPLAIN-Informationen, die in einem EXPLAIN-Exemplar erfasst werden, schließen die Umgebung der SQL-Kompilierung und den zur Ausführung der SQL-Anweisung, die kompiliert wird, ausgewählten Zugriffsplan mit ein. Zum Beispiel kann ein EXPLAIN-Exemplar aus jeder der folgenden Informationsgruppen bestehen:

- Alle wählbaren SQL-Anweisungen in einem Paket für statische SQL-Anweisungen. Sie können EXPLAIN-Informationen für Anweisungen mit SELECT, SELECT INTO, UPDATE, INSERT, VALUES, VALUES INTO und DELETE erfassen.
- Eine bestimmte SQL-Anweisung für SQL-Anweisungen zum inkrementellen Binden (Incremental Bind).
- Eine bestimmte SQL-Anweisung für dynamische SQL-Anweisungen.
- Jede SQL-Anweisung EXPLAIN (dynamisch oder statisch).

Die Informationen der EXPLAIN-Tabellen spiegeln die Beziehungen zwischen Operatoren und Datenobjekten im Zugriffsplan wieder. Die folgende Abbildung zeigt die Beziehungen zwischen diesen Tabellen.

EXPLAIN-Informationen werden in den folgenden Tabellen gespeichert:

Tabelle 30. Relationale Tabellen zur Speicherung von EXPLAIN-Daten

Tabellenname	Beschreibung
EXPLAIN_ARGUMENT	Enthält die spezifischen Merkmale für jeden einzelnen Operator (sofern vorhanden).
EXPLAIN_INSTANCE	Die Hauptsteuertabelle für alle EXPLAIN-Informationen. Jede Datenzeile in den EXPLAIN-Tabellen ist explizit mit einer eindeutigen Zeile in dieser Tabelle verbunden. In dieser Tabelle werden grundlegende Informationen über die Quelle der SQL-Anweisungen, die mit EXPLAIN bearbeitet werden, und Umgebungsinformationen gespeichert.
EXPLAIN_OBJECT	Identifiziert die Datenobjekte, die für den Zugriffsplan erforderlich sind, der zur Erfüllung der SQL-Anweisung generiert wurde.
EXPLAIN_OPERATOR	Enthält alle Operatoren, die zur Erfüllung der SQL-Anweisung durch den SQL-Compiler benötigt werden.
EXPLAIN_PREDICATE	Enthält Informationen über die Vergleichselemente, die von einem bestimmten Operator angewandt werden.
EXPLAIN_STATEMENT	Enthält den Text der SQL-Anweisung, wie er für die verschiedenen Stufen der EXPLAIN-Informationen vorhanden ist. Die ursprüngliche, vom Benutzer eingegebene SQL-Anweisung wird in dieser Tabelle mit der vom Optimierungsprogramm zum Auswählen eines Zugriffsplans verwendeten Version gespeichert. Wenn eine EXPLAIN-Momentaufnahme (Snapshot) angefordert wird, werden zusätzliche EXPLAIN-Informationen aufgezeichnet, um den Zugriffsplan zu beschreiben, der vom SQL-Optimierungsprogramm ausgewählt wurde. Diese Informationen werden in der Spalte SHAPSHOT der Tabelle EXPLAIN_STATEMENT in dem für Visual Explain erforderlichen Format gespeichert. Dieses Format kann von anderen Anwendungen nicht verwendet werden.
EXPLAIN_STREAM	Stellt die Eingabe- und Ausgabedatenströme zwischen einzelnen Operatoren und Datenobjekten dar. Die Datenobjekte selbst werden in der Tabelle EXPLAIN_OBJECT dargestellt. Die an einem Datenstrom beteiligten Operatoren werden in der Tabelle EXPLAIN_OPERATOR dargestellt.
ADVISE_WORKLOAD	Ermöglicht Benutzern, eine Auslastung für die Datenbank zu beschreiben. Jede Zeile in der Tabelle stellt eine SQL-Anweisung in der Auslastung dar und wird durch eine zugeordnete Häufigkeit beschrieben. Das Tool `db2advis` verwendet diese Tabelle, um Auslastungsdaten zu erfassen und zu speichern.
ADVISE_INSTANCE	Enthält Informationen zur Ausführung von `db2advis`. Diese Informationen beinhalten auch die Startzeit. Für jede Ausführung von `db2advis` wird eine Zeile eingefügt.
ADVISE_INDEX	Speichert Informationen über empfohlene Indizes. Die Tabelle kann durch den SQL-Compiler, das Dienstprogramm `db2advis` oder einen Benutzer mit Daten gefüllt werden. Diese Tabelle wird zu zwei Zwecken verwendet: • Zum Abrufen empfohlener Indizes • Zum Bewerten von Indizes auf der Grundlage von Eingaben zu vorgeschlagenen Indizes
ADVISE_MQT	Enthält die CREATE-DDL-Anweisungen, die Abfrage, die die einzelnen gespeicherten Abfragetabellen (MQT) definiert, die Statistikdaten für die gespeicherten Abfragetabellen wie beispielsweise COLSTATS (pro Spalte) im XML-Format, NUMROWS usw. sowie die Stichprobenabfrage zum Abrufen von Stichprobenstatistiken für die gespeicherten Abfragetabellen.

Tabelle 30. Relationale Tabellen zur Speicherung von EXPLAIN-Daten (Forts.)

Tabellenname	Beschreibung
ADVISE_TABLE	Speichert die DDL-Anweisungen zur Tabellenerstellung mit den endgültigen Empfehlungen des Designadvisors für empfohlene gespeicherte Abfragetabellen, MDC-Tabellen und Partitionierung abhängig von den angegebenen Optionen und den generierten Empfehlungen.
ADVISE_PARTITION	Speichert von `db2advis` generierte und bewertete virtuelle Partitionen.

Anmerkung: Nicht alle der oben genannten Tabellen werden standardmäßig erstellt. Zur Erstellung der Tabellen führen Sie die Prozedur EXPLAIN.DDL aus, die sich im Unterverzeichnis *misc* des Unterverzeichnisses *sqllib* befindet.

EXPLAIN-Tabellen können für mehrere Benutzer gemeinsame Daten enthalten. Jedoch können die EXPLAIN-Tabellen auch für nur einen Benutzer definiert werden. Anschließend können Aliasnamen für jeden weiteren Benutzer definiert werden, der den gleichen Namen verwendet, um auf die definierten Tabellen zu verweisen. Jeder Benutzer, der auf die gemeinsamen EXPLAIN-Tabellen zugreift, muss das Zugriffsrecht INSERT zum Einfügen für diese Tabellen haben.

Zugehörige Konzepte:

- „SQL-EXPLAIN-Einrichtung" auf Seite 223
- „EXPLAIN-Informationen für Datenobjekte" auf Seite 229
- „EXPLAIN-Informationen für Exemplare" auf Seite 231
- „EXPLAIN-Informationen für Datenoperatoren" auf Seite 230
- „SQL-EXPLAIN-Tools" auf Seite 637

EXPLAIN-Informationen für Datenobjekte

Ein einzelner Zugriffsplan kann zur Ausführung der SQL-Anweisung ein oder mehrere Datenobjekte verwenden.

Objektstatistiken: Die EXPLAIN-Einrichtung zeichnet Informationen über das Objekt auf, wie z. B. folgende Angaben:

- Die Erstellungszeit
- Den Zeitpunkt, an dem zum letzten Mal Statistiken für das Objekt gesammelt wurden
- Eine Angabe, ob die Daten im Objekt sortiert sind (nur Tabellen- oder Indexobjekte)
- Die Anzahl von Spalten im Objekt (nur Tabellen- oder Indexobjekte)
- Die geschätzte Anzahl von Zeilen im Objekt (nur Tabellen- oder Indexobjekte)
- Die Anzahl von Seiten, die das Objekt im Pufferpool einnimmt
- Den geschätzten Gesamtaufwand in Millisekunden für eine einzelne wahlfreie Ein-/Ausgabeoperation für den angegebenen Tabellenbereich, in dem dieses Objekt gespeichert ist
- Die geschätzte Übertragungsrate in Millisekunden zum Lesen einer 4-KB-Seite aus dem angegebenen Tabellenbereich
- Die Größen für PREFETCHSIZE und EXTENTSIZE (in 4-KB-Seiten)
- Den Grad der Datenclusterbildung in Bezug auf den Index

- Die Anzahl von Blattseiten (Leaf Pages), die vom Index für dieses Objekt verwendet werden, und die Anzahl der Indexstufen in der Indexbaumstruktur
- Die Anzahl unterschiedlicher vollständiger Schlüsselwerte im Index für dieses Objekt
- Die Gesamtzahl der Überlaufsätze in der Tabelle

Zugehörige Konzepte:
- „Die EXPLAIN-Tabellen und die Organisation von EXPLAIN-Informationen" auf Seite 227
- „EXPLAIN-Informationen für Exemplare" auf Seite 231
- „EXPLAIN-Informationen für Datenoperatoren" auf Seite 230
- „Richtlinien zur Analyse von EXPLAIN-Informationen" auf Seite 236

EXPLAIN-Informationen für Datenoperatoren

Ein einziger Zugriffsplan kann mehrere Operationen mit den Daten ausführen, um die SQL-Anweisung auszuführen und die Ergebnisse an Sie zurückzugeben. Der SQL-Compiler ermittelt, welche Operationen erforderlich sind, wie zum Beispiel eine Tabellensuche, eine Indexsuche, eine Verknüpfung über Verschachtelungsschleife oder ein GROUP BY-Operator.

Neben den Operatoren, die in einem Zugriffsplan verwendet werden, und den Informationen zu jedem einzelnen Operator zeigen EXPLAIN-Informationen auch die kumulativen Auswirkungen des Zugriffsplans.

Informationen über den geschätzten Aufwand: Für die Operatoren können die folgenden geschätzten kumulativen Aufwände angezeigt werden. Diese Aufwände beziehen sich auf den ausgewählten Zugriffsplan bis zu dem Operator (einschließlich), für den die Informationen erfasst wurden.
- Der Gesamtaufwand (in Timerons)
- Die Anzahl der Seiten-E/As
- Die Anzahl von CPU-Instruktionen
- Der Aufwand (in Timerons) für das Abrufen der ersten Zeile einschließlich eines evtl. zusätzlich erforderlichen einleitenden Systemaufwands
- Der Übertragungsaufwand (in Rahmen (Frames))

Timerons ist die Bezeichnung für eine künstlich geschaffene relative Maßeinheit. Timerons werden vom Optimierungsprogramm auf der Basis interner Werte bestimmt, zum Beispiel Statistikdaten, die sich aufgrund der Datenbanknutzung ändern. Daher gibt es keine Garantie, dass das Timerons-Maß für eine SQL-Anweisung bei jeder Ermittlung der geschätzten Kosten in Timerons gleich ist.

Operatoreigenschaften: Die folgenden Informationen werden von der EXPLAIN-Einrichtung aufgezeichnet, um die Eigenschaften jedes Operators zu beschreiben:
- Die Menge der Tabellen, auf die zugegriffen wurde
- Die Menge der Spalten, auf die zugegriffen wurde
- Die Spalten, nach denen die Daten sortiert werden, wenn das Optimierungsprogramm ermittelt hat, dass diese Sortierung von nachfolgenden Operatoren verwendet werden kann
- Die Menge der Vergleichselemente, die angewandt wurden
- Die geschätzte Anzahl von Zeilen, die übergeben werden sollen (Kardinalität)

Zugehörige Konzepte:

- „Die EXPLAIN-Tabellen und die Organisation von EXPLAIN-Informationen" auf Seite 227
- „EXPLAIN-Informationen für Datenobjekte" auf Seite 229
- „EXPLAIN-Informationen für Exemplare" auf Seite 231
- „Richtlinien zur Analyse von EXPLAIN-Informationen" auf Seite 236

EXPLAIN-Informationen für Exemplare

Informationen über EXPLAIN-Exemplare werden in der Tabelle EXPLAIN_INSTANCE gespeichert. Zusätzliche spezifische Informationen zu jeder SQL-Anweisung in einem Exemplar werden in der Tabelle EXPLAIN_STATEMENT gespeichert.

Kennzeichnung von EXPLAIN-Exemplaren: Anhand der folgenden Informationen können Sie jedes EXPLAIN-Exemplar identifizieren und die Informationen für SQL-Anweisungen einem bestimmten Aufruf der Einrichtung zuordnen:

- Der Benutzer, der die EXPLAIN-Informationen anforderte
- Der Zeitpunkt, zu dem die EXPLAIN-Anforderung begann
- Der Name des Pakets, das die mit EXPLAIN bearbeitete SQL-Anweisung enthält
- Das Schema des Pakets, das die mit EXPLAIN bearbeitete SQL-Anweisung enthält
- Die Version des Pakets, das die Anweisung enthält
- Eine Angabe, ob Momentaufnahmendaten (Snapshot) erfasst wurden

Einstellungen der Umgebung: Es werden Informationen zur Umgebung des Datenbankmanagers erfasst, in der der SQL-Compiler die Abfragen optimiert hat. Zu den Umgebungsinformationen gehören die folgenden:

- Die Versions- und Releasenummer für die Stufe von DB2®
- Der Grad der Parallelität, für den die Abfrage kompiliert wurde

  Das Sonderregister CURRENT DEGREE, die Bindeoption DEGREE, die API SET RUNTIME DEGREE und der Konfigurationsparameter *dft_degree* bestimmen den Grad der Parallelität, für den eine bestimmte Abfrage kompiliert wird.
- Die Angabe, ob es sich um eine dynamische oder statische SQL-Anweisung
- Die zum Kompilieren der Abfrage verwendete Optimierungsklasse
- Der Typ der Zeilenblockung für Cursor, der beim Kompilieren der Abfrage angegeben wurde
- Die Isolationsstufe, in der die Abfrage ausgeführt wird
- Die Werte verschiedener Konfigurationsparameter zum Zeitpunkt der Kompilierung der Abfrage. Die folgenden Parameter werden bei der Erfassung einer EXPLAIN-Momentaufnahme (Snapshot) aufgezeichnet:
  - Zwischenspeicher für Sortierlisten (*sortheap*)
  - Durchschnittliche Anzahl aktiver Anwendungen (*avg_appls*)
  - Datenbankzwischenspeicher (*dbheap*)
  - Maximaler Speicher für Sperrenliste (*locklist*)
  - Maximale Anzahl Sperren pro Anwendung (*maxlocks*)
  - CPU-Geschwindigkeit (*cpuspeed*)
  - Kommunikationsbandbreite (*comm_bandwidth*)

Kennzeichnung von SQL-Anweisungen: Für jedes EXPLAIN-Exemplar können mehrere SQL-Anweisungen mit EXPLAIN bearbeitet werden. Zusätzlich zu den Informationen, die das EXPLAIN-Exemplar eindeutig kennzeichnen, kann jede einzelne SQL-Anweisung anhand der folgenden Informationen identifiziert werden:

- Der Typ der Anweisung: SELECT, DELETE, INSERT, UPDATE, positioniertes DELETE, positioniertes UPDATE
- Die Anweisungs- und Abschnittsnummer des Pakets, das die SQL-Anweisung absetzt, wie in der Katalogsicht SYSCAT.STATEMENTS gespeichert

Die Felder QUERYTAG und QUERYNO in der Tabelle EXPLAIN_STATEMENT enthalten Kennungen, die im Rahmen des EXPLAIN-Prozesses mit Werten gefüllt werden. Für dynamische SQL-EXPLAIN-Anweisungen, die in einer Sitzung des Befehlszeilenprozessors (CLP) oder einer Sitzung von Call Level Interface (CLI) übergeben wurden, wird als Wert für QUERYTAG „CLP" bzw. „CLI" gespeichert, wenn EXPLAIN MODE oder EXPLAIN SNAPSHOT aktiv ist. In diesem Fall wird für QUERYNO standardmäßig eine Nummer angegeben, die mindestens um den Wert 1 für jede Anweisung erhöht wird. Für alle anderen dynamischen SQL-EXPLAIN-Anweisungen, die nicht über CLP bzw. CLI angefordert werden oder die SQL-EXPLAIN-Anweisung nicht verwenden, wird das Feld QUERYTAG mit Leerzeichen und das QUERYNO immer mit dem Wert „1" gefüllt.

Schätzung des Aufwands: Für jede mit EXPLAIN bearbeitete Anweisung zeichnet das Optimierungsprogramm einen Schätzwert für den relativen Aufwand zur Ausführung des ausgewählten Zugriffsplans auf. Dieser Aufwand wird in einer künstlich geschaffenen relativen Maßeinheit mit der Bezeichnung *timeron* angegeben. Schätzwerte für die benötigten Ausführungszeiten werden aus folgenden Gründen nicht bereitgestellt:

- Das SQL-Optimierungsprogramm schätzt nicht die benötigte Zeit, sondern nur den Ressourcenbedarf ab.
- Das Optimierungsprogramm modelliert nicht alle Faktoren nach, die die benötigte Zeit beeinflussen können. Es ignoriert Faktoren, die keine Auswirkung auf die Effizienz des Zugriffsplans haben. Die benötigte Zeit wird von einer Reihe Laufzeitfaktoren beeinflusst, zu denen die folgenden gehören: die Systemauslastung, die Ressourcenverfügbarkeit, der Umfang der Parallelverarbeitung und der Ein-/Ausgabeoperationen, der Aufwand für die Rückgabe von Zeilen an den Benutzer sowie die Übertragungszeit zwischen Client und Server.

Anweisungstext: Für jede mit EXPLAIN bearbeitete Anweisung werden zwei Versionen des Texts der SQL-Anweisung aufgezeichnet. Eine Version ist der Code, den der SQL-Compiler von der Anwendung empfängt. Die andere Version ist die rückübersetzte Version aus der compilerinternen Darstellung der Abfrage. Obwohl diese Rückübersetzung anderen SQL-Anweisungen sehr ähnlich sieht, entspricht sie nicht unbedingt der richtigen SQL-Syntax und spiegelt nicht in jedem Fall den tatsächlichen Inhalt der internen Darstellung als Ganzes wider. Diese Übersetzung wird nur zur Verfügung gestellt, um Ihnen einen Einblick in den SQL-Kontext zu geben, in dem das Optimierungsprogramm den Zugriffsplan ausgewählt hat. Um zu verstehen, wie der SQL-Compiler Ihre Abfrage zur Optimierung umgeschrieben hat, vergleichen Sie den vom Benutzer geschriebenen Anweisungstext mit der internen Darstellung der SQL-Anweisung. Die umgeschriebene Anweisungen zeigt Ihnen außerdem noch weitere Elemente in der Umgebung, die sich auf Ihre Anweisung auswirken, wie zum Beispiel Auslöser und Integritätsbedingungen. Einige Schlüsselwörter, die in diesem „optimierten" Text verwendet werden, sind:

$C*n* Der Name einer abgeleiteten Spalte, wobei *n* ein ganzzahliger Wert ist.

$CONSTRAINT$	Dieses Kennzeichen markiert den Namen einer Integritätsbedingung, die der ursprünglichen SQL-Anweisung bei der Kompilierung hinzugefügt wurde. Es tritt in Kombination mit dem Präfix $WITH_CONTEXT$ auf.
$DERIVED.T*n*	Der Name einer abgeleiteten Tabelle, wobei *n* ein ganzzahliger Wert ist.
$INTERNAL_FUNC$	Dieses Kennzeichen markiert das Vorhandensein einer Funktion, die vom SQL-Compiler für die mit EXPLAIN bearbeitete Abfrage verwendet wurde, jedoch nicht zur allgemeinen Verwendung verfügbar ist.
$INTERNAL_PRED$	Dieses Kennzeichen markiert das Vorhandensein eines Vergleichselements, das vom SQL-Compiler bei der Kompilierung der mit EXPLAIN bearbeiteten Abfrage hinzugefügt wurde, jedoch nicht zur allgemeinen Verwendung verfügbar ist. Vom Compiler wird ein internes Vergleichselement verwendet, um den infolge von Auslösern und Integritätsbedingungen der ursprünglichen SQL-Anweisung hinzugefügten Bedingungskontext zu erfüllen.
RID	Dieses Kennzeichen dient zur Identifizierung der Spalte der Satz-ID (RID) für eine bestimmte Zeile.
$TRIGGER$	Dieses Kennzeichen markiert den Namen eines Auslösers, der der ursprünglichen SQL-Anweisung bei der Kompilierung hinzugefügt wurde. Es tritt in Kombination mit dem Präfix $WITH_CONTEXT$ auf.
$WITH_CONTEXT$(...)	Dieses Präfix tritt am Anfang des Texts auf, wenn zusätzliche Auslöser oder Integritätsbedingungen in die ursprüngliche SQL-Anweisung eingefügt wurden. Diesem Präfix folgt eine Liste der Namen aller Auslöser oder Integritätsbedingungen, die sich auf die Kompilierung und Auflösung der SQL-Anweisung auswirken.

Zugehörige Konzepte:

- „Die EXPLAIN-Tabellen und die Organisation von EXPLAIN-Informationen" auf Seite 227
- „EXPLAIN-Informationen für Datenobjekte" auf Seite 229
- „EXPLAIN-Informationen für Datenoperatoren" auf Seite 230
- „Richtlinien zur Analyse von EXPLAIN-Informationen" auf Seite 236

Zugehörige Referenzen:

- „comm_bandwidth - Kommunikationsbandbreite" auf Seite 529
- „sortheap - Zwischenspeichergröße für Sortierlisten" auf Seite 418
- „locklist - Maximaler Speicher für Sperrenliste" auf Seite 399
- „maxlocks - Maximale Anzahl Sperren pro Anwendung" auf Seite 433
- „dbheap - Zwischenspeicher für Datenbank" auf Seite 399
- „cpuspeed - CPU-Geschwindigkeit" auf Seite 530

- „avg_appls - Durchschnittliche Anzahl aktiver Anwendungen” auf Seite 443
- „dft_degree - Standardgrad der Parallelität” auf Seite 502

Richtlinien zur Erfassung von EXPLAIN-Informationen

EXPLAIN-Daten werden bei der Kompilierung einer SQL-Anweisung erfasst, wenn Sie dies anfordern. Bei der Anforderung von EXPLAIN-Daten sollten Sie berücksichtigen, wie die erfassten Informationen später verwendet werden sollen.

Anmerkungen:

1. Wenn SQL-Anweisungen zum inkrementellen Binden (Incremental Bind) bei der Ausführung kompiliert werden, werden Daten bei der Ausführung, nicht bei der Bindeoperation, in die EXPLAIN-Tabellen eingetragen. Für solche Anweisungen sind das Qualifikationsmerkmal und die Berechtigungs-ID, die in die EXPLAIN-Tabelle eingetragen werden, die des Paketeigners und nicht die des Benutzers, der das Paket ausführt.
2. Die EXPLAIN-Informationen werden nur erfasst, wenn die SQL-Anweisung kompiliert wird. Nach der Erstkompilierung werden dynamische SQL-Anweisungen erneut kompiliert, wenn eine Änderung an der Umgebung dies erfordert oder wenn die EXPLAIN-Einrichtung aktiv ist. Wenn Sie dieselbe Anweisung PREPARE mehrmals für die gleiche SQL-Anweisung ausführen, erfolgt das Kompilieren der SQL-Anweisung und das Erfassen der EXPLAIN-Daten jedes Mal, wenn diese Anweisung vorbereitet oder ausgeführt wird.
3. Wenn ein Paket mit der Bindeoption REOPT ONCE/ALWAYS gebunden wird, werden SQL-Anweisungen, die Hostvariablen, Parametermarken oder Sonderregister enthalten, mit den realen Werten dieser Variablen, wenn sie bekannt sind, und mit Standardschätzwerten, wenn die Werte nicht bekannt sind, beim Kompilieren kompiliert und der Zugriffspfad wird erstellt.
4. Wenn die Klausel FOR REOPT ONCE verwendet wird, wird ein Versuch unternommen, die angegebene SQL-Anweisung mit der gleichen Anweisung im Paketcache abzugleichen. Die Werte dieser bereits reoptimierten SQL-Anweisung im Cache werden zur Reoptimierung der angegebenen SQL-Anweisung verwendet. Die EXPLAIN-Tabellen enthalten in diesem Fall den neu generierten, reoptimierten Zugriffsplan und die Werte, die für diese Reoptimierung verwendet wurden, sofern der Benutzer über die erforderlichen Zugriffsrechte verfügt.
5. In einem System mit mehreren Partitionen muss die Anweisung in der gleichen Partition mit EXPLAIN bearbeitet werden, in der sie auch ursprünglich kompiliert und mit REOPT ONCE reoptimiert wurde. Ansonsten wird ein Fehler zurückgegeben.

Erfassen von Informationen in den EXPLAIN-Tabellen

- **Statische SQL-Anweisungen oder SQL-Anweisungen zum inkrementellen Binden**:

  Geben Sie in den Befehlen BIND oder PREP die Optionen EXPLAIN ALL oder EXPLAIN YES an oder fügen Sie eine statische SQL-Anweisung EXPLAIN in das Quellenprogramm ein.

- **Dynamische SQL-Anweisungen**:

  Informationen für EXPLAIN-Tabellen werden in folgenden Fällen erfasst:

  – Das Sonderregister CURRENT EXPLAIN MODE enthält einen der folgenden Werte:

    - YES: Der SQL-Compiler erfasst EXPLAIN-Daten und führt die SQL-Anweisung aus.

- EXPLAIN: Der SQL-Compiler erfasst EXPLAIN-Daten, führt die SQL-Anweisung jedoch nicht aus.
- RECOMMEND INDEXES: Der SQL-Compiler erfasst EXPLAIN-Daten und die Daten zu den empfohlenen Indizes werden in der Tabelle ADVISE_INDEX gespeichert, jedoch wird die SQL-Anweisung nicht ausgeführt.
- EVALUATE INDEXES: Der SQL-Compiler verwendet die vom Benutzer in die Tabelle ADVISE_INDEX eingefügten Indizes zur Beurteilung verwendet. Im Modus EVALUATE INDEXES werden alle dynamischen Anweisungen mit EXPLAIN so bearbeitet, als wären diese virtuellen Indizes verfügbar. Der SQL-Compiler wählt in diesem Fall die virtuellen Indizes, wenn sie die Leistung der Anweisungen verbessern. Ansonsten werden die Indizes ignoriert. Durch eine Analyse der EXPLAIN-Ergebnisse können Sie feststellen, ob die vorgeschlagenen Indizes nützlich wären.

- Die Option EXPLAIN ALL wurde im Befehl BIND oder PREP angegeben. Der SQL-Compiler erfasst EXPLAIN-Daten für dynamisches SQL bei der Ausführung, selbst wenn das Sonderregister CURRENT EXPLAIN MODE auf den Wert NO gesetzt ist. Die SQL-Anweisung wird ausgeführt und gibt die Abfrageergebnisse zurück.

Erfassen von EXPLAIN-Momentaufnahmen

Wenn eine EXPLAIN-Momentaufnahme (Snapshot) angefordert wird, werden EXPLAIN-Informationen in der Spalte SNAPSHOT der Tabelle EXPLAIN_STATEMENT in dem für Visual Explain erforderlichen Format gespeichert. Dieses Format kann von anderen Anwendungen nicht verwendet werden. Weitere Informationen zum Inhalt der Informationen der EXPLAIN-Momentaufnahmen erhalten Sie direkt mit Hilfe von Visual Explain. Zu diesen Informationen gehören Informationen zu Datenobjekten und Datenoperatoren.

EXPLAIN-Momentaufnahmedaten werden erfasst, wenn eine SQL-Anweisung kompiliert wird und EXPLAIN-Daten wie folgt angefordert wurden:

- **Statische SQL-Anweisungen oder SQL-Anweisungen zum inkrementellen Binden**:

  Eine EXPLAIN-Momentaufnahme wird erfasst, wenn eine der beiden Klauseln EXPLSNAP ALL oder EXPLSNAP YES in den Befehlen BIND oder PREP angegeben wird oder wenn das Quellenprogramm eine statische SQL-Anweisung EXPLAIN mit der Klausel FOR SNAPSHOT oder WITH SNAPSHOT enthält.

- **Dynamische SQL-Anweisungen**:

  Eine EXPLAIN-Momentaufnahme wird in folgenden Fällen erfasst:

  – Sie führen eine SQL-Anweisung EXPLAIN mit der Klausel FOR SNAPSHOT oder WITH SNAPSHOT aus. Mit der Klausel FOR SNAPSHOT werden nur Momentaufnahmedaten erfasst. Mit der Klausel WITH SNAPSHOT werden neben den Momentaufnahmedaten auch sämtliche anderen EXPLAIN-Informationen erfasst.

  – Das Sonderregister CURRENT EXPLAIN SNAPSHOT enthält einen der folgenden Werte:
    - YES: Der SQL-Compiler erfasst EXPLAIN-Momentaufnahmedaten und führt die SQL-Anweisung aus.
    - EXPLAIN: Der SQL-Compiler erfasst EXPLAIN-Momentaufnahmedaten, führt die SQL-Anweisung jedoch nicht aus.

  – Sie geben die Option EXPLSNAP ALL im Befehl BIND oder PREP an. Der SQL-Compiler erfasst EXPLAIN-Momentaufnahmedaten bei der Ausführung,

selbst wenn das Sonderregister CURRENT EXPLAIN SNAPSHOT auf den Wert NO eingestellt ist. Die SQL-Anweisung wird ausgeführt.

Zugehörige Konzepte:

- „SQL-EXPLAIN-Einrichtung" auf Seite 223
- „Richtlinien zur Verwendung von EXPLAIN-Informationen" auf Seite 226
- „Die EXPLAIN-Tabellen und die Organisation von EXPLAIN-Informationen" auf Seite 227
- „Der Designadvisor" auf Seite 237
- „Richtlinien zur Analyse von EXPLAIN-Informationen" auf Seite 236
- „SQL-EXPLAIN-Tools" auf Seite 637

Richtlinien zur Analyse von EXPLAIN-Informationen

Obwohl der Hauptzweck der EXPLAIN-Informationen in der Analyse der Zugriffspläne für SELECT-Anweisungen liegt, kann Ihnen die Analyse der EXPLAIN-Daten auf verschiedene Weisen helfen, Ihre Abfragen und Ihre Umgebung zu optimieren. Betrachten Sie die folgende Art einer Analyse:

- **Verwendung von Indizes**

  Die geeigneten Indizes können die Leistung erheblich fördern. Anhand der EXPLAIN-Ausgabe können Sie ermitteln, ob die von Ihnen für eine bestimmte Gruppe von Abfragen erstellten Indizes tatsächlich verwendet werden. In der EXPLAIN-Ausgabe sollten Sie folgende Bereiche auf die Verwendung von Indizes hin überprüfen:
  - Verknüpfungsvergleichselemente
  - Lokale Vergleichselemente
  - Die Klausel GROUP BY
  - Die Klausel ORDER BY
  - Die SELECT-Liste

  Sie können die EXPLAIN-Einrichtung auch verwenden, um zu ermitteln, ob ein anderer Index anstelle des vorhandenen Index oder kein Index verwendet werden könnte. Führen Sie nach der Erstellung eines neuen Index den Befehl RUNSTATS aus, um Statistikdaten für diesen Index zu erfassen, und kompilieren Sie die Abfrage erneut. Mit der Zeit werden Sie möglicherweise durch die EXPLAIN-Daten feststellen, dass anstelle einer Indexsuche eine Tabellensuche verwendet wird. Dies kann sich aus einer Änderung in der Clusterbildung der Tabellendaten ergeben. Wenn der zuvor verwendete Index nun ein niedriges Clusterverhältnis aufweist, sollten Sie in Erwägung ziehen, die Tabelle zu reorganisieren, um ihre Daten in Clustern entsprechend dem Index anzuordnen, den Befehl RUNSTATS auszuführen, um Statistikdaten für Index und Tabelle zu erfassen, und anschließend die Abfrage erneut zu kompilieren. Prüfen Sie die EXPLAIN-Ausgabe erneut, um herauszufinden, ob das Reorganisieren der Tabelle zu einer Verbesserung des Zugriffsplans geführt hat.

- **Zugriffstyp**

  Analysieren Sie die EXPLAIN-Ausgabe und suchen Sie nach Arten des Zugriffs auf die Daten, die für die Art von Anwendung, die Sie ausführen, normalerweise nicht optimal sind. Zum Beispiel:

  - **OLTP-Abfragen (Online Transaction Processing)**

    In OLTP-Anwendungen werden häufig Indexsuchen mit Vergleichselementen durchgeführt, die eine Bereichsbegrenzung vornehmen, weil diese Anwendungen in der Regel nur wenige Zeilen zurückgeben, die den mit einem Gleichheitsvergleichselement für eine Schlüsselspalte angegebenen Bedingun-

gen entsprechen. Wenn Ihre OLTP-Abfragen eine Tabellensuche verwenden, können Sie die EXPLAIN-Daten analysieren, um herauszufinden, warum keine Indexsuche verwendet wurde.

– **Reine Suchabfragen**

  Die Suchbedingungen für eine Abfrage, bei der Daten nur abgerufen werden, können sehr vage sein, was jedoch bewirkt, dass eine große Menge von Zeilen den Bedingungen entspricht. Wenn sich Benutzer normalerweise nur einige Seiten der Ausgabedaten anzeigen lassen, können Sie dafür sorgen, dass nicht die gesamte Antwortmenge errechnet werden muss, bevor einige Ergebnisse zurückgegeben werden. In diesem Fall unterscheiden sich die Ziele des Benutzers vom grundlegenden Verarbeitungsprinzip des Optimierungsprogramms, das versucht, den Ressourcenbedarf für die gesamte Abfrage und nicht nur für die ersten wenigen Anzeigen mit Daten zu minimieren.

  Wenn zum Beispiel die EXPLAIN-Ausgabe zeigt, dass Operatoren sowohl für Mischverknüpfungen als auch für Sortierungen im Zugriffsplan verwendet wurden, wird die gesamte Antwortmenge in einer temporären Tabelle gespeichert, bevor Zeilen an die Anwendung zurückgegeben werden. In diesem Fall können Sie versuchen, den Zugriffsplan durch die Verwendung der Klausel OPTIMIZE FOR in der SELECT-Anweisung zu ändern. Wenn Sie diese Option angeben, kann das Optimierungsprogramm versuchen, einen Zugriffsplan auszuwählen, der nicht die gesamte Antwortmenge in einer temporären Tabelle erstellt, bevor die ersten Zeilen an die Anwendung zurückgegeben werden.

- **Verknüpfungsmethoden**

  Wenn bei einer Abfrage zwei Tabellen verknüpft werden, sollten Sie die Art der verwendeten Verknüpfungsverarbeitung überprüfen. Verknüpfungen mit relativ vielen Zeilen, wie sie zum Beispiel bei Abfragen von Entscheidungshilfedaten auftreten, werden in der Regel schneller als Mischverknüpfungen (Merge Joins) ausgeführt. Verknüpfungen, an denen nur einige wenige Zeilen beteiligt sind, wie zum Beispiel für OLTP-Abfragen, sind zumeist als Verknüpfungen über Verschachtelungsschleife (Nested Loop Join) effizienter. In beiden Fällen kann es jedoch auch Umstände geben, wie beispielsweise die Verwendung lokaler Vergleichselemente oder Indizes, die die normale Verarbeitung dieser Verknüpfungen möglicherweise ändern.

Zugehörige Konzepte:

- „SQL-EXPLAIN-Einrichtung" auf Seite 223
- „SQL-EXPLAIN-Tools" auf Seite 637
- „Beschreibung der Ausgabe von db2expln und dynexpln" auf Seite 645

Der Designadvisor

Der Designadvisor (DB2® Design Advisor) ist ein Tool, das Ihnen helfen kann, die Auslastungsleistung Ihres Systems erheblich zu verbessern. Die Aufgabe zu entscheiden, welche Indizes, gespeicherte Abfragetabellen (MQTs) oder Clusteringdimensionen für eine komplexe Auslastung zu erstellen sind, kann sich enorm kompliziert 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 Abfragetabellen (MQTs)

- Umwandlung in MDC-Tabellen (mit mehrdimensionalem Clustering)
- Umpartitionierung von Tabellen
- Löschen von Indizes und gespeicherten Abfragetabellen, 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.

Über die grafische Benutzerschnittstelle (GUI) des Designadvisors bzw. das Befehlszeilentool können Sie den Designadvisor zur Vereinfachung der folgenden Aufgaben einsetzen:

Planen oder Einrichten einer neuen Datenbank

Beim Entwurf Ihrer Datenbank können Sie den Designadvisor zu folgenden Zwecken einsetzen:

- Generieren von Entwurfsalternativen in einer Testumgebung für eine partitionierte Datenbankumgebung sowie für Indizes, gespeicherte Abfragetabellen (MQTs) und MDC-Tabellen.
- Für partitionierte Datenbankumgebungen kann der Designadvisor mit folgenden Zielen verwendet werden:
  - Ermitteln der Partitionierungsstrategie vor dem Laden von Daten in eine Datenbank
  - Unterstützen der Migration von einer DB2-Einzelpartitionsdatenbank auf eine DB2-Datenbank mit mehreren Partitionen
  - Unterstützen der Migration von einem anderen Datenprodukt auf eine DB2-Datenbank mit mehreren Partitionen
- Bewerten von Indizes, gespeicherten Abfragetabellen, MDC-Tabellen oder Partitionierungsstrategien, die manuell generiert wurden.

Optimieren der Auslastungsleistung

Nach der Einrichtung Ihrer Datenbank können Sie den Designadvisor zu folgenden Zwecken einsetzen:

- Verbessern der Leistung einer bestimmten Anweisung oder Auslastung
- Verbessern der allgemeinen Datenbankleistung unter Verwendung der Leistung einer Musterauslastung als Messgröße
- Verbessern der Leistung der am häufigsten ausgeführten Abfragen, wie sie zum Beispiel durch den Aktivitätsmonitor ermittelt werden
- Bestimmen, wie sich die Leistung einer neuen kritischen Abfrage optimieren lässt
- Reagieren auf Empfehlungen der Diagnosezentrale im Hinblick auf Probleme mit dem gemeinsamen Dienstprogrammspeicher oder dem Sortierspeicher bei einer sortierintensiven Auslastung
- Ermitteln von Objekten, die in keiner Auslastung verwendet werden

Ausgabe des Designadvisors

Über die grafische Benutzerschnittstelle (GUI) des Designadvisors können Sie Empfehlungen aus dem Designadvisor anzeigen, speichern oder implementieren. Wenn Sie den Designadvisor über die Befehlszeile ausführen, wird die Ausgabe standardmäßig an die Stanardausgabeeinheit (stdout) gesendet und in den Tabellen ADVISE_TABLE und ADVISE_INDEX gespeichert.

- Die Tabelle ADVISE_TABLE enthält USE_TABLE='Y' für MQT- und MDC-Tabellen sowie Empfehlungen zu Partitionierungsstrategien.

  Die Empfehlungen zu gespeicherten Abfragetabellen (MQT) sind außerdem in der Tabelle ADVISE_MQT, die MDC-Empfehlungen in der Tabelle ADVISE_TABLE und die Empfehlungen zu Partitionierungsstrategien in der Tabelle ADVISE_PARTITION zu finden. Der Wert RUN_ID in diesen Tabellen entspricht dem Wert START_TIME in der Tabelle ADVISE_INSTANCE für jede Ausführung des Designadvisors.
- Die Tabelle ADVISE_INDEX enthält den Wert USE_INDEX='Y' oder 'R' für Indexempfehlungen.

Die Tabelle ADVISE_INSTANCE wird ebenfalls mit einer Zeile pro Ausführung des Designadvisors aktualisiert:

- Das Feld START_TIME und das Feld END_TIME zeigen die Start- bzw. Stoppzeiten des Dienstprogramms.
- In das Feld STATUS wird der Wert 'COMPLETED' eingetragen, wenn das Dienstgramm erfolgreich beendet wird.
- Das Feld MODE zeigt an, ob die Option -m verwendet wurde.
- Das Feld COMPRESSION gibt den Typ der verwendeten Komprimierung an.

Sie können die Empfehlungen des Designadvisors über die Option -o in einer Datei speichern. Die gespeicherte Ausgabe des Designadvisors besteht aus den folgenden Elementen:

- Die CREATE-Anweisungen, die für neue Indizes, gespeicherte Abfragetabellen, Partitionierungsstrategien und MDC-Tabellen angegeben werden
- REFRESH-Anweisungen für gespeicherte Abfragetabellen
- RUNSTATS-Befehle für neue Objekte
- Empfohlene, bereits vorhandene gespeicherte Abfragetabellen, sofern sie zur Ausführung der Auslastung verwendet wurden und werden

  Anmerkung: Die Spalte COLSTATS der Tabelle ADVISE_MQT enthält die Spaltenstatistiken für eine gespeicherte Abfragetabelle. Die Statistiken besitzen eine XML-Struktur wie die folgende:

```
<?xml version=\"1.0\" encoding=\"USASCII\"?>
<colstats>
    <column>
         <name>SPALTENNAME1</name>
         <colcard>1000</colcard>
         <high2key>999</high2key>
         <low2key>2</low2key>
    </column>
    ....

    <column>
         <name>SPALTENNAME100</name>
         <colcard>55000</colcard>
         <high2key>49999</high2key>
         <low2key>100</low2key>
    </column>
</colstats>
```

  Beachten Sie, dass die XML-Struktur mehr als eine Spalte beinhalten kann. Für jede Spalte wird die Spaltenkardinalität (d. h. die Anzahl der Werte in der Spalte) sowie optional die Werte für HIGH2KEY und LOW2KEY angegeben.

Nach einigen kleineren Modifikationen können Sie diese Ausgabedatei als Prozedur für den Befehlszeilenprozessor (CLP) zur Erstellung der empfohlenen Objekte verwenden. Folgende Modifikationen bieten sich eventuell an:

- Kombinieren aller RUNSTATS-Befehlsanweisungen zu einem einzigen RUNSTATS-Aufruf für die neuen oder modifizierten Objekte
- Angeben geeigneterer Objektnamen als die systemgenerierten IDs
- Entfernen aller DDL-Anweisungen für Objekte, die nicht sofort implementiert werden sollen, aus dem Code durch Löschen oder Verwenden von Kommentarzeichen

Zugehörige Konzepte:

- „Datenpartitionierung" in *Systemverwaltung: Konzept*
- „Vor- und Nachteile von Indizes" auf Seite 288
- „Indexplanung - Tipps" auf Seite 290
- „Gespeicherte Abfragetabellen" auf Seite 207
- „Tabellen mit mehrdimensionalem Clustering" in *Systemverwaltung: Konzept*

Zugehörige Referenzen:

- „db2advis - DB2 Design Advisor Command" in *Command Reference*

Definieren einer Auslastung für den Designadvisor

Eine *Auslastung* ist eine Gruppe von SQL-Anweisungen, die der Datenbankmanager in einem bestimmten Zeitraum verarbeiten muss. Zum Beispiel könnte der Datenbankmanager in einem Monat 1000 INSERT-, 10000 UPDATE-, 10000 SELECT- und 1000 DELETE-Anweisungen verarbeiten müssen. Der Designadvisor analysiert eine angegebene Auslastung und zieht Faktoren, wie den Typ der Auslastungsanweisungen, die Häufigkeit, mit der eine bestimmte Anweisung auftritt, sowie Merkmale Ihrer Datenbank in Betracht, um Empfehlungen zu generieren, die den Gesamtaufwand zur Ausführung der Auslastung minimieren.

Vorgehensweise:

Über die Designadvisor-GUI:

Über die Seite „Auslastung" der grafischen Benutzerschnittstelle des Designadvisors können Sie eine neue Auslastungsdatei erstellen oder eine bereits vorhandene Auslastungsdatei modifizieren. Sie können Anweisungen in die Datei aus verschiedenen Quellen importieren:

- Eine Textdatei mit Begrenzern
- Eine Ereignismonitortabelle
- Query Patroller-Protokolldatentabellen durch die Option -qp über die Befehlszeile
- Mit EXPLAIN bearbeitete Anweisungen in der Tabelle EXPLAINED_STATEMENT
- Neue SQL-Anweisungen, die mit einer DB2-Momentaufnahme erfasst wurden

Nach dem Importieren Ihrer SQL-Anweisungen können Sie Anweisungen hinzufügen, ändern, modifizieren oder entfernen sowie die Häufigkeit der Anweisungen modifizieren.

Über die Befehlszeile:

Über die Befehlszeile führen Sie den Designadvisor wie folgt aus:

- Mit einer einzigen SQL-Anweisung, die Sie in der Befehlzeile zusammen mit dem Befehl angeben
- Mit einer Gruppe dynamischer SQL-Anweisungen, die in einer DB2-Momentaufnahme erfasst wurden
- Mit einer Gruppe von SQL-Anweisungen, die in einer Auslastungsdatei enthalten sind

Gehen Sie wie folgt vor, um den Designadvisor für dynamische SQL-Anweisungen auszuführen:

1. Setzen Sie den Datenbankmonitor mit dem folgenden Befehl zurück:

   ```
   db2 reset monitor for database datenbankname
   ```

2. Warten Sie einen geeigneten Zeitraum für die Ausführung dynamischer SQL-Anweisungen für die Datenbank ab.
3. Führen Sie den Befehl **db2advis** mit der Option -g aus. Wenn Sie die dynamischen SQL-Anweisungen in der Tabelle ADVISE_WORKLOAD zur späteren Referenz speichern wollen, verwenden Sie außerdem die Option -p.

Gehen Sie wie folgt vor, um den Designadvisor für eine Gruppe von SQL-Anweisungen auszuführen, die in einer Auslastungsdatei enthalten sind:

1. Erstellen Sie manuell eine Auslastungsdatei, indem Sie jede SQL-Anweisung mit einem Semikolon trennen, oder importieren Sie SQL-Anweisungen aus einer oder mehreren der oben aufgeführten Quellen.
2. Legen Sie die Häufigkeit der Anweisungen in der Auslastung fest. Jeder Anweisung in der Auslastungsdatei wird standardmäßig die Häufigkeit 1 zugeordnet. Die Häufigkeit einer SQL-Anweisung stellt die Anzahl der Vorkommen der Anweisung innerhalb der Auslastung im Verhältnis zur Anzahl der Vorkommen anderer Anweisungen dar. Eine bestimmte SELECT-Anweisung könnte zum Beispiel 100-mal in einer Auslastung vorkommen, während eine andere SELECT-Anweisung 10-mal vorkommt. Zur Darstellung der relativen Häufigkeit dieser beiden Anweisungen würden Sie der ersten SELECT-Anweisung eine Häufigkeit von 10 und der zweiten SELECT-Anweisung eine Häufigkeit von 1 zuordnen. Sie können die Häufigkeit bzw. die Wertigkeit einer bestimmten Anweisung in der Auslastung manuell ändern, indem Sie die folgende Zeile nach der Anweisung einfügen: - - # SET FREQUENCY *n*. Dabei ist *n* der Häufigkeitswert, den Sie der Anweisung zuordnen wollen.
3. Geben Sie den Befehl **db2advis** mit der Option -i gefolgt vom Namen der Auslastungsdatei ein.

Zur Ausführung des Designadvisors für eine Auslastung, die in der Tabelle ADVISE_WORKLOAD enthalten ist, geben Sie den Befehl **db2advis** mit der Option -w gefolgt vom Namen der Auslastung ein.

Zugehörige Konzepte:

- „Der Designadvisor" auf Seite 237

Verwenden des Designadvisors zur Migration von einer Datenbank mit einer einzelnen Partition auf eine Datenbank mit mehreren Partitionen

Sie können den Designadvisor zur Unterstützung einer Migration von einer Datenbank mit einer Einzelpartition auf eine Datenbank mit mehreren Partitionen verwenden. Der Designadvisor kann Ihnen Empfehlungen für die Partitionierung Ihrer Daten und gleichzeitig Empfehlungen für neue Indizes, gespeicherte Abfragetabellen (MQTs) und MDC-Tabellen (mit mehrdimensionalem Clustering) geben.

Vorgehensweise:

1. Aktualisieren Sie den Produktlizenzschlüssel für DB2 UDB ESE.
2. Erstellen Sie mindestens einen Tabellenbereich in einer Partitionsgruppe einer Datenbank mit mehreren Partitionen.

   Anmerkung: Da der Designadvisor nur eine Partitionierung für vorhandene Tabellenbereiche empfehlen kann, müssen die Tabellenbereiche, die der Designadvisor in Betracht ziehen soll, in der partitionierten Datenbank vorhanden sein, bevor der Designadvisor ausgeführt wird.
3. Führen Sie den Designadvisor mit ausgewählter Partitionierungsfunktion über die grafische Benutzerschnittstelle oder unter Angabe der Partitionierungsoption mit dem Befehl **db2advis** aus.
4. Wenn Sie mit dem Designadvisor in der Steuerzentrale arbeiten, können Sie die Partitionierungsempfehlungen automatisch implementieren. Wenn Sie den Befehl **db2advis** verwenden, müssen Sie die **db2advis**-Ausgabedatei geringfügig modifizieren, bevor Sie die vom Designadvisor generierten DDL-Anweisungen ausführen.

Zugehörige Konzepte:

- „Der Designadvisor" auf Seite 237

Zugehörige Tasks:

- „Registrieren der DB2-Produktlizenzberechtigung mit dem Befehl db2licm" in *Installation und Konfiguration Ergänzung*

Begrenzungen und Einschränkungen des Designadvisors

1. **Einschränkungen für Indexempfehlungen**
   - Indizes, die für gespeicherte Abfragetabellen (MQTs) empfohlen werden, dienen zur Verbesserung der Auslastungsleistung, und nicht der REFRESH TABLE-Leistung. Wenn außerdem Aktualisierungs-, Einfüge- und Löschoperationen kein Bestandteil der Auslastung sind, würde die Leistung einer Änderung (z. B. Aktualisierung) der gespeicherten Abfragetabelle für mit IMMEDIATE definierte Abfragetabellen nicht berücksichtigt. Aus diesem Grund ist es empfehlenswert, für solche gespeicherten Abfragetabellen eindeutige Indizes für den von der mit IMMEDIATE definierten gespeicherten Abfragetabelle implizierten eindeutigen Schlüssel zu haben. Der implizierte eindeutige Schlüssel basiert auf den Spalten in der GROUP BY-Klausel der Definition der gespeicherten Abfragetabelle.
   - Der RID-Clusterindex wird nur empfohlen, wenn mehrdimensionales Clustering auszuwählen ist. Der Designadvisor schließt RID-Clusterindizes als Option anstelle der Erstellung einer MDC-Struktur für eine Tabelle mit ein.

2. **Einschränkungen für MQT-Empfehlungen**
   - Der Designadvisor empfiehlt keine inkrementellen gespeicherten Abfragetabellen (MQTs). Wenn Sie inkrementelle gespeicherte Abfragetabellen erstellen wollen, können Sie mit REFRESH IMMEDIATE definierte gespeicherte Abfragetabellen nehmen und diese in inkrementelle Abfragetabellen mit einer eigenen Auswahl an Zwischenspeichertabellen konvertieren.
   - Indizes, die für gespeicherte Abfragetabellen (MQTs) empfohlen werden, dienen zur Verbesserung der Auslastungsleistung, und nicht der REFRESH-Leistung für gespeicherte Abfragetabellen.
   - Wenn Aktualisierungs-, Einfüge- und Löschoperationen kein Bestandteil der angegebenen Auslastung sind, wird der Leistungsaufwand zur Aktualisierung einer empfohlenen, mit REFRESH IMMEDIATE definierten gespeicherten Abfragetabelle nicht berücksichtigt. Es wird empfohlen, für gespeicherte Abfragetabellen mit REFRESH IMMEDIATE eindeutige Indizes für den implizierten eindeutigen Schlüssel zu erstellen, der sich aus den Spalten in der GROUP BY-Klausel der Definition der gespeicherten Abfragetabelle zusammensetzt.
3. **Einschränkungen für MDC-Empfehlungen**
   - Vorhandene Tabellen müssen Daten enthalten. Ansonsten wird MDC für die Tabelle nicht in Betracht gezogen.
   - MDC-Empfehlungen für neue gespeicherte Abfragetabellen werden nicht in Betracht gezogen, sofern nicht die Stichprobenoption -r mit dem Befehl verwendet wird oder die MQT-Stichprobenerstellung im GUI-Tool ausgewählt wird.
   - Der Designadvisor gibt keine MDC-Empfehlungen für typisierte oder temporäre Tabellen.
   - Der Designadvisor gibt keine MDC-Empfehlungen für zusammengeschlossene Tabellen.
   - Der Speicherplatz für die Stichprobendaten, der während der Ausführung des Designadvisors verwendet wird, muss vorhanden sein. Ansonsten wird die Stichprobentabelle nur für Basisspalten unter der Annahme unkorrelierter Daten untersucht. In diesem Fall wird eine Warnung generiert.
   - Tabellen ohne erfasste Statistiken werden bei der Analyse übersprungen.
   - Der Designadvisor gibt keine Empfehlungen für mehrspaltige Dimensionen.
   - Vorhandene Tabellen müssen Daten enthalten, damit die Stichprobenerstellung bei der MDC-Auswahl funktioniert.
4. **Einschränkungen für Partitionierungsempfehlungen**

   Der Designadvisor kann eine Partitionierung nur unter DB2® Enterprise Server Edition empfehlen. Wenn die Partitionierungsoptionen mit dem Befehl **db2advis** angegeben werden, wird ein Fehler zurückgegeben. In der grafischen Benutzerschnittstelle (GUI) des Designadvisors ist die Partitionierungsfunktion in einer Datenbankumgebung mit nur einer Partition nicht auswählbar.
5. **Weitere Einschränkungen**

   Während der Ausführung des Designadvisors werden Simulationskatalogtabellen erstellt. Diese Tabellen werden gelöscht, wenn die Ausführung des Designadvisors beendet ist. Eine unvollständige Ausführung des Designadvisors kann dazu führen, dass einige dieser Tabellen nicht gelöscht werden.

In diesem Fall können Sie den Designadvisor dazu verwenden, diese Tabellen zu löschen, indem Sie das Dienstprogramm über die Befehlszeile erneut starten. Zum Entfernen der Simulationskatalogtabellen geben Sie die Option -f und die Option -n an (geben Sie mit -n den gleichen Benutzernamen ein, der bei der unvollständigen Ausführung verwendet wurde). Wenn Sie die Option -f nicht angeben, generiert der Designadvisor die DROP-Anweisungen, die zum Entfernen der Tabellen benötigt werden.

Sie sollten einen getrennten Tabellenbereich zum Speichern dieser simulierten Katalogtabellen erstellen und DROP TABLE RECOVERY auf den Wert OFF setzen. Dies ermöglicht eine einfachere Bereinigung und eine schnellere Ausführung des Designadvisors.

Zugehörige Konzepte:

- „Der Designadvisor" auf Seite 237

Teil 3. Optimieren und Konfigurieren des Systems

Kapitel 8. Leistung bei der Ausführung

Dieses Kapitel enthält Informationen zu Faktoren, die sich auf die Leistung von SQL-Abfragen zur Laufzeit auswirken. Außerdem kann es sinnvoll sein, die Informationen zu Aspekten des physischen Datenbankentwurfs, insbesondere in Bezug auf die Vorteile einer Partitionierung, einer Verwendung von MDC-Tabellen mit mehrdimensionalem Clustering und ähnlichen Einrichtungen hinzuzuziehen.

Speichernutzung

Dieses Kapitel beschreibt, wie der Datenbankmanager den Speicher nutzt, und listet die Parameter auf, mit denen die Speichernutzung durch den Datenbankmanager und Datenbanken gesteuert werden kann.

Organisation der Speichernutzung

Kenntnisse darüber, wie DB2® den Speicher organisiert, helfen Ihnen bei der Feinabstimmung der Speichernutzung zur Erzielung einer guten Leistung. Zahlreiche Konfigurationsparameter wirken sich auf die Speichernutzung aus. Einige betreffen den Speicher auf dem Server, andere den Speicher auf dem Client und wieder andere sowohl den Speicher auf dem Server als auch den Speicher auf dem Client. Darüber hinaus wird Speicher zu verschiedenen Zeiten und aus verschiedenen Bereichen des Systems zugeordnet und wieder freigegeben. Während der Datenbankserver aktiv ist, können Sie die Größe von Speicherbereichen im gemeinsamen Datenbankspeicher ändern.

Ein Systemadministrator muss auf eine ausgewogene Gesamtnutzung der Speicherkapazität auf dem System achten. Verschiedene Anwendungen, die unter dem Betriebssystem aktiv sind, können den Speicher auf unterschiedliche Weise nutzen. Obwohl einige Anwendungen zum Beispiel den Cache des Betriebssystems verwenden können, nutzt der Datenbankmanager nicht den Cache des Betriebssystems, sondern einen eigenen Pufferpool zum Zwischenspeichern von Daten.

Die folgende Abbildung zeigt die unterschiedlichen Teile des Speichers, den der Datenbankmanager für verschiedene Nutzungen zuordnet.

Anmerkung: Die Abbildung zeigt nicht, wie der Speicher in einer Enterprise Server Edition-Umgebung verwendet wird, die mehrere logische Knoten umfasst. In einer solchen Umgebung enthält jeder Knoten einen gemeinsamen Datenbankspeichersatz.

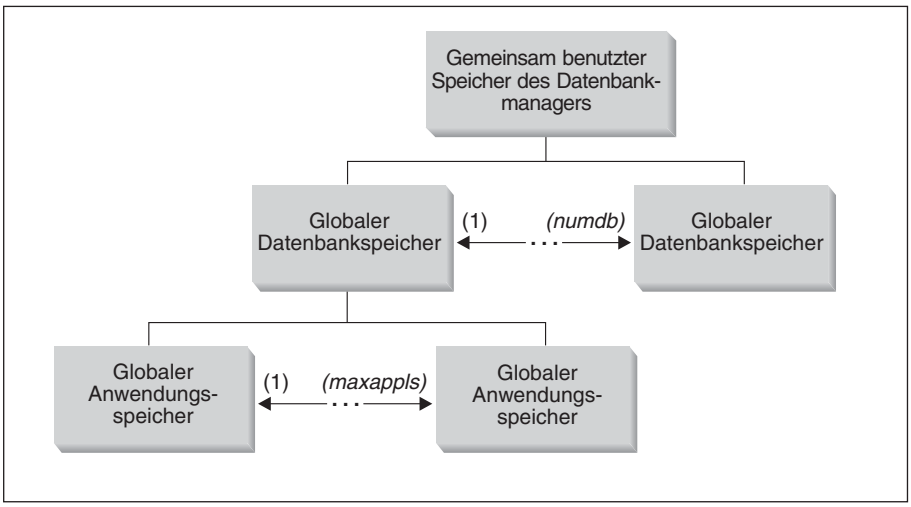


Abbildung 19. Arten von Speicher, die vom Datenbankmanager verwendet werden

Der Speicher wird für jedes Exemplar des Datenbankmanagers zugeordnet, wenn die folgenden Ereignisse eintreten:

- **Wenn der Datenbankmanager gestartet wird (`db2start`):** Der globale gemeinsame Speicher des Datenbankmanagers wird zugeordnet und bleibt zugeordnet, bis der Datenbankmanager gestoppt wird (`db2stop`). Dieser Bereich enthält Informationen, die der Datenbankmanager zur Verwaltung von Aktivitäten für alle Datenbankverbindungen verwendet. Wenn die erste Anwendung die Verbindung zu einer Datenbank herstellt, werden sowohl globale als auch private Speicherbereiche zugeordnet.
- **Wenn eine Datenbank aktiviert oder zum ersten Mal eine Verbindung zu ihr hergestellt wird:** Der globale Datenbankspeicher wird zugeordnet. Der globale Datenbankspeicher wird für alle Anwendungen verwendet, die eine Verbindung zur Datenbank herstellen. Die Größe des globalen Datenbankspeichers wird durch den Konfigurationsparameter *database_memory* festgelegt. Sie können zu Anfang mehr Speicher als benötigt angeben, so dass der zusätzliche Speicher später dynamisch verteilt werden kann. Obwohl die Gesamtgröße des globalen Datenbankspeichers nicht geändert werden kann, während die Datenbank aktiv ist, können Bereiche angepasst werden, die im globalen Datenbankspeicher enthalten sind. Solche Bereiche sind zum Beispiel Pufferpools, die Sperrenliste, der Datenbankzwischenspeicher und der Zwischenspeicher für Dienstprogramme sowie der Paketcache und der Katalogcache. In einer Umgebung, in der der Konfigurationsparameter für partitionsinterne Parallelität (intra_parallel) des Datenbankmanagers aktiviert ist, oder einer Umgebung, in der der Verbindungskonzentrator aktiviert ist, wird auch der gemeinsame Sortierspeicher als Teil des globalen Datenbankspeichers zugeordnet.
- **Wenn eine Anwendung die Verbindung zu einer Datenbank herstellt:** In einer partitionierten Datenbankumgebung, in einer nicht partitionierten Datenbank bei aktiviertem Konfigurationsparameter für partitionsinterne Parallelität (*intra_parallel*) des Datenbankmanagers oder in einer Umgebung mit aktiviertem Verbindungskonzentrator können mehrere Anwendungen *Anwendungsgruppen* zugeordnet werden, um Speicher gemeinsam zu nutzen. Jede Anwendungsgruppe verfügt über eine eigene Zuordnung an gemeinsamem Speicher. In dem gemeinsamen Speicher der Anwendungsgruppe verfügt jede Anwendung über einen eigenen Zwischenspeicher zur Anwendungssteuerung, nutzt jedoch den gemeinsamen Zwischenspeicher der Anwendungsgruppe.

Die folgenden drei Datenbankkonfigurationsparameter bestimmen die Größe des Speichers von Anwendungsgruppen:

- Der Parameter *appgroup_mem_sz*, der die Größe des gemeinsamen Speichers für die Anwendungsgruppe angibt
- Der Parameter *groupheap_ratio*, der den Prozentsatz des gemeinsamen Anwendungsgruppenspeichers angibt, der für den gemeinsamen Zwischenspeicher bereitgestellt wird
- Der Parameter *app_ctl_heap_sz*, der die Größe des Steuerungszwischenspeichers für jede Anwendung in der Gruppe angibt

Der Leistungsvorteil einer Gruppierung des Anwendungsspeichers liegt in der besseren Effizienz der Cache- und Speichernutzung.

Einige Elemente des globalen Anwendungsspeichers können auch dynamisch in der Größe angepasst werden.

- **Wenn ein Agent erstellt wird:** Dieses Ereignis wird in der Abbildung nicht gezeigt. Der private Agentenspeicher wird für einen Agenten zugeordnet, wenn der Agent infolge einer Verbindungsanforderung oder einer neuen SQL-Anforderung in einer parallelen Umgebung zugeordnet wird. Der private Agentenspeicher wird für den Agenten zugeordnet und enthält Speicherbereiche, die nur von diesem bestimmten Agenten verwendet werden, wie zum Beispiel den Sortierspeicher und den Anwendungszwischenspeicher.

  Wenn bereits eine Datenbank von einer Anwendung verwendet wird, werden für Anwendungen, die nachfolgend eine Verbindung herstellen, nur der private Agentenspeicher und der globale gemeinsame Anwendungsspeicher zugeordnet.

Die Abbildung gibt außerdem die folgenden Konfigurationsparameter an, die die Größe des Speichers begrenzen, der für die einzelnen Zwecke zugeordnet wird. Beachten Sie, dass dieser Speicher in einer partitionierten Datenbankumgebung in jeder Datenbankpartition zugeordnet wird.

- *numdb*

  Dieser Parameter 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 möglicherweise zugeordnet wird, wenn Sie den Wert dieses Parameters erhöhen.

- *maxappls*

  Dieser Parameter definiert die maximale Anzahl von Anwendungen, die gleichzeitig mit einer einzigen Datenbank verbunden sein können. Er wirkt sich auf die Menge an Speicher aus, die möglicherweise für privaten Agentenspeicher und globalen Anwendungsspeicher für diese Datenbank zugeordnet wird. Beachten Sie, dass dieser Parameter für jede Datenbank unterschiedlich eingestellt werden kann.

- *maxagents* und *max_coordagents* bei Parallelverarbeitung

  Diese Parameter werden in der Abbildung nicht gezeigt. Sie begrenzen die Anzahl von Datenbankmanageragenten, die gleichzeitig in allen aktiven Datenbanken in einem Exemplar vorhanden sein können. Zusammen mit dem Parameter *maxappls* begrenzen diese Parameter die Größe des Speichers, der für den privaten Agentenspeicher und den globalen Anwendungsspeicher zugeordnet wird.

Zugehörige Konzepte:

- „Gemeinsam benutzter Speicher des Datenbankmanagers" auf Seite 250
- „Der FCM-Pufferpool und Speicheranforderungen" auf Seite 252
- „Globaler Speicher und zugehörige Steuerparameter" auf Seite 253
- „Richtlinien zur Optimierung von Parametern mit Auswirkung auf die Speichernutzung" auf Seite 255
- „Speicherverwaltung" auf Seite 38

Gemeinsam benutzter Speicher des Datenbankmanagers

Für die Ausführung des Datenbankmanagers ist Speicher erforderlich. Dieser Speicher kann, insbesondere in Umgebungen mit partitionsinterner und partitionsübergreifender Parallelität, sehr groß sein.

Die folgende Abbildung zeigt, wie Speicher zur Unterstützung von Anwendungen verwendet wird. Die gezeigten Konfigurationsparameter ermöglichen Ihnen, die Größe dieses Speichers zu steuern, indem Sie die Anzahl von Speichersegmenten (d. h. logischen Speicherbereichen) und ihre Größe begrenzen.

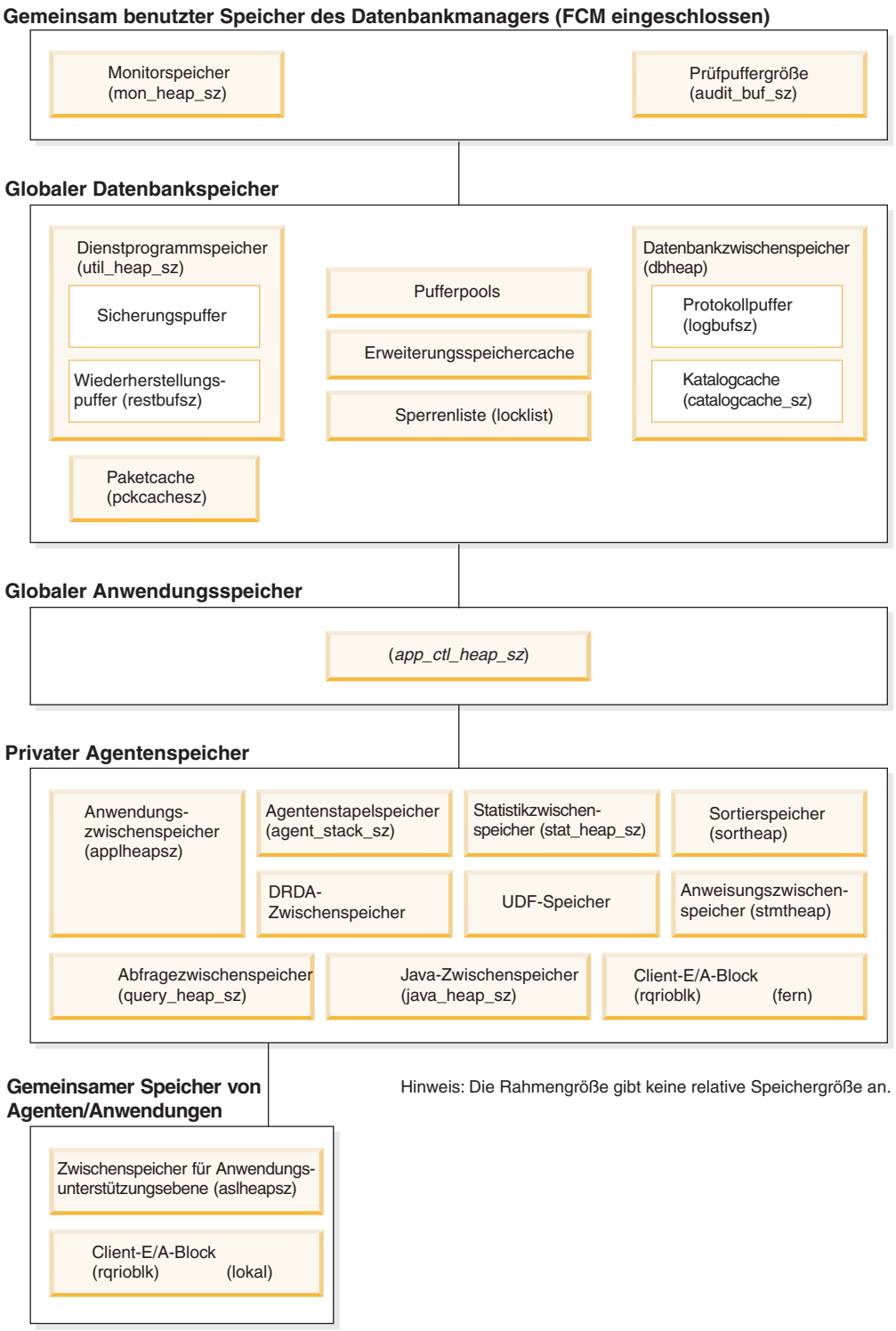


Abbildung 20. Verwendung von Speicher durch den Datenbankmanager

Anhand der Informationen über Datenbankagenten können Sie die Größe dieses Speichers abschätzen und steuern. Agenten, die für Anwendungen aktiv sind, benötigen einen beträchtlichen Speicherbereich, insbesondere wenn der Wert des Parameters *maxagents* zu hoch eingestellt ist. Dies liegt daran, dass eine gewisse Speichergröße für jeden Agenten gemäß dem Wert des Parameters *maxagents* vorab zugeordnet wird. Wenn der Parameter *maxagents* auf einen höheren Wert als nötig

gesetzt ist, wird einiger Speicher vorab zugeordnet, jedoch vom Datenbankmanager gar nicht genutzt, so dass unnötig Speicher verschwendet wird.

Bei partitionierten Datenbanksystemen benötigt FCM (Fast Communications Manager) einen beträchtlichen Speicherbereich, insbesondere wenn der Wert des Parameters *fcm_num_buffers* hoch eingestellt ist. Zudem wird der benötigte FCM-Speicher entweder aus dem FCM-Pufferpool oder aus dem gemeinsamen Speicher des Datenbankmanagers und dem FCM-Pufferpool zugeordnet, je nachdem, ob ein partitioniertes Datenbanksystem mehrere logische Knoten verwendet oder nicht.

Zugehörige Konzepte:

- „Organisation der Speichernutzung" auf Seite 247
- „Globaler Speicher und zugehörige Steuerparameter" auf Seite 253
- „Pufferpoolverwaltung" auf Seite 257
- „Richtlinien zur Optimierung von Parametern mit Auswirkung auf die Speichernutzung" auf Seite 255
- „Speicherverwaltung" auf Seite 38

Der FCM-Pufferpool und Speicheranforderungen

Wenn Sie ein partitioniertes Datenbanksystem haben, das nicht mehrere logische Knoten besitzt, werden der gemeinsame Speicher des Datenbankmanagers und der FCM-Pufferpool wie in folgender Abbildung gezeigt eingesetzt.

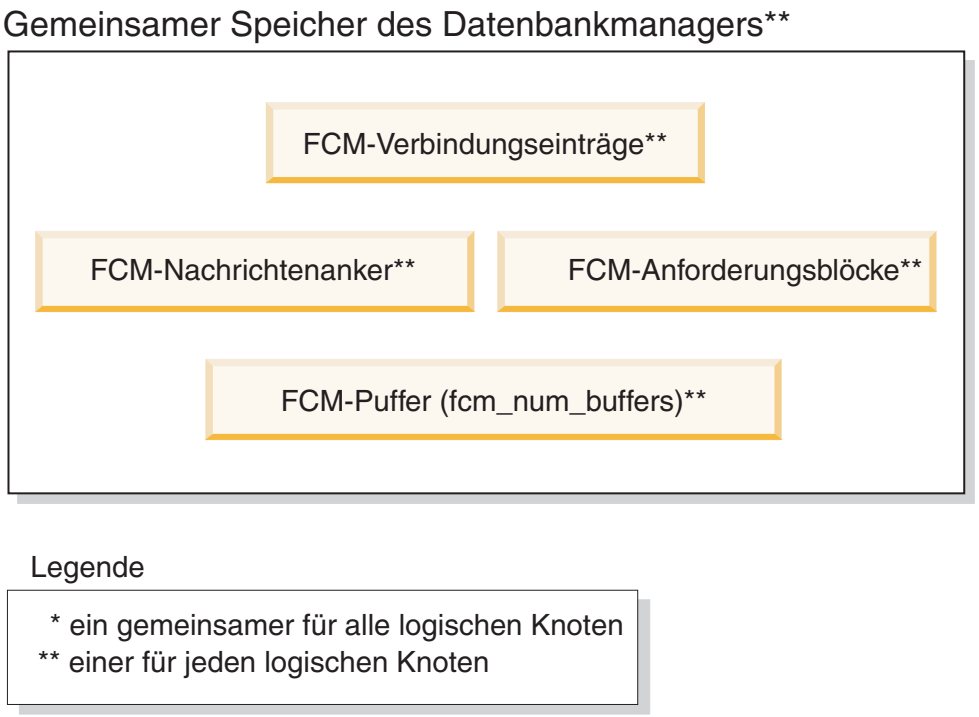


Abbildung 21. FCM-Pufferpool ohne Verwendung mehrerer logischer Knoten

Wenn Sie ein partitioniertes Datenbanksystem haben, das über mehrere logische Knoten verfügt, werden der gemeinsame Speicher des Datenbankmanagers und der FCM-Pufferpool wie folgt genutzt.

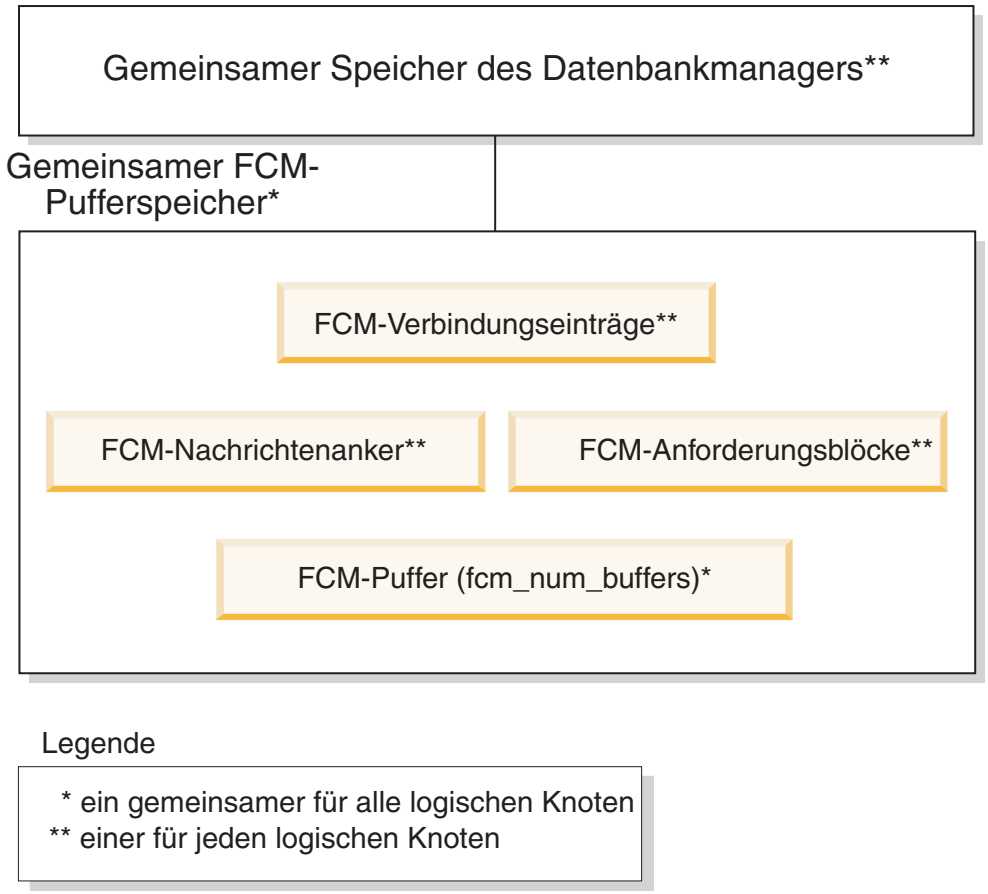


Abbildung 22. FCM-Pufferpool bei Verwendung mehrerer logischer Knoten

Beginnen Sie zur Konfiguration des FCM (Fast Communications Manager) mit dem Standardwert für die Anzahl von FCM-Puffern (*fcm_num_buffers*). Weitere Informationen zu FCM auf AIX®-Plattformen finden Sie in der Beschreibung der Registriervariablen DB2_FORCE_FCP_BP.

Überwachen Sie zur Optimierung dieses Parameters den erreichten unteren Grenzwert für die freien Puffer mit Hilfe des Datenbanksystemmonitors.

Zugehörige Konzepte:

- „Gemeinsam benutzter Speicher des Datenbankmanagers" auf Seite 250

Globaler Speicher und zugehörige Steuerparameter

Der gemeinsame Datenbankmanagerspeicher besteht aus den folgenden Komponenten:

Globaler Datenbankspeicher

Der globale Datenbankspeicher wird durch folgende Konfigurationsparameter beeinflusst:

- Der Parameter *database_memory* stellt einen unteren Grenzwert für die Größe des globalen Datenbankspeichers bereit.
- Die folgenden Parameter oder Faktoren geben die maximale Größe von Speichersegmenten an:
  - Die Größe der Pufferpools.

- Maximaler Speicher für Sperrenliste (*locklist*)
- Datenbankzwischenspeicher (*dbheap*)
- Zwischenspeichergröße für Dienstprogramme (*util_heap_sz*)
- Segmentgröße für erweiterten Speicher (*estore_seg_sz*)
- Segmentanzahl für erweiterten Speicher (*num_estore_segs*)
- Größe des Paketcache (*pckcachesz*)
- Schwellenwert für Sortierspeicher für gemeinsame Sortiervorgänge (*sheapthres_shr*)

Globaler Anwendungsspeicher

Der globale Anwendungsspeicher wird durch den Konfigurationsparameter für die Größe des Zwischenspeichers zur Anwendungssteuerung (*app_ctl_heap_sz*) beeinflusst.

Für parallele Systeme ist außerdem Speicherbereich für den Zwischenspeicher für Anwendungssteuerung erforderlich, der von den Agenten gemeinsam benutzt wird, die für dieselbe Anwendung in einer Datenbankpartition aktiv sind. Der Zwischenspeicher wird zugeordnet, wenn der erste Agent, der eine Anforderung von der Anwendung empfängt, eine Verbindung anfordert. Der Agent kann entweder ein koordinierender Agent oder ein Subagent sein.

Privater Agentenspeicher

- Die Anzahl von Speichersegmenten wird durch den niedrigeren der folgenden Werte begrenzt:
  - Die Summe des Konfigurationsparameters *maxappls* für alle aktiven Datenbanken, die die maximal zulässige Anzahl aktiver Anwendungen angibt
  - Der Wert des Konfigurationsparameters *maxagents*, der die maximal zulässige Anzahl von Agenten angibt
- Die maximale Größe von Speichersegmenten wird von den Werten der folgenden Parameter bestimmt:
  - Zwischenspeichergröße für Anwendungen (*applheapsz*)
  - Zwischenspeicher für Sortierlisten (*sortheap*)
  - Größe des Anweisungszwischenspeichers (*stmtheap*)
  - Größe des Statistikzwischenspeichers (*stat_heap_sz*)
  - Größe des Abfragezwischenspeichers (*query_heap_sz*)
  - Größe des Agentenstacks (*agent_stack_sz*)

Gemeinsamer Agenten-/Anwendungsspeicher

- Die Gesamtanzahl der Segmente für gemeinsamen Agenten-/Anwendungsspeicher für lokale Clients wird durch den niedrigeren der folgenden Datenbankkonfigurationsparameter begrenzt:
  - Die Summe von *maxappls* für alle aktiven Datenbanken
  - Der Wert des Parameters *maxagents* bzw. des Parameters *max_coordagents* für Parallelsysteme
- Der gemeinsame Agenten-/Anwendungsspeicher wird außerdem durch folgende Datenbankkonfigurationsparameter beeinflusst:
  - Zwischenspeichergröße für Anwendungsunterstützungsebene (*aslheapsz*)
  - E/A-Blockgröße für Clients (*rqrioblk*)

Zugehörige Referenzen:

- „estore_seg_sz - Segmentgröße für erweiterten Speicher" auf Seite 438
- „max_coordagents - Maximale Anzahl koordinierender Agenten" auf Seite 445
- „num_estore_segs - Segmentanzahl für erweiterten Speicher" auf Seite 438
- „sortheap - Zwischenspeichergröße für Sortierlisten" auf Seite 418
- „maxagents - Maximale Anzahl von Agenten" auf Seite 446
- „aslheapsz - Zwischenspeichergröße für Anwendungsunterstützungsebene" auf Seite 421
- „applheapsz - Zwischenspeichergröße für Anwendungen" auf Seite 412
- „pckcachesz - Größe des Paketcache" auf Seite 403
- „locklist - Maximaler Speicher für Sperrenliste" auf Seite 399
- „rqrioblk - E/A-Blockgröße für Clients" auf Seite 424
- „dbheap - Zwischenspeicher für Datenbank" auf Seite 399
- „stmtheap - Größe des Anweisungszwischenspeichers" auf Seite 420
- „maxappls - Maximale Anzahl aktiver Anwendungen" auf Seite 447
- „agent_stack_sz - Größe des Agentenstacks" auf Seite 411
- „query_heap_sz - Größe des Abfragezwischenspeichers" auf Seite 415
- „util_heap_sz - Zwischenspeichergröße für Dienstprogramme" auf Seite 406
- „stat_heap_sz - Größe des Statistikzwischenspeichers" auf Seite 419
- „database_memory - Größe des gemeinsam benutzten Datenbankspeichers" auf Seite 398

Richtlinien zur Optimierung von Parametern mit Auswirkung auf die Speichernutzung

Die erste Regel zur Einstellung von Parametern zur Speicherzuordnung ist, sie nie auf die höchsten Werte setzen, sofern ein solcher Wert nicht sorgfältig auf seine Eignung geprüft wurde. Diese Regel gilt selbst für Systeme, die über maximale Speicherkapazitäten verfügen. Viele Parameter, die sich auf den Speicher auswirken, können zulassen, dass der Datenbankmanager leicht und schnell den gesamten auf einem Computer verfügbaren Speicher belegt. Darüber hinaus kann die Verwaltung einer großen Speichermenge wesentlich mehr Zusatzaufwand auf Seiten des Datenbankmanagers verursachen, so dass der Systemaufwand steigt.

Einige UNIX®-Betriebssysteme ordnen Auslagerungsspeicher zu, wenn ein Prozess Speicher zuordnet, und nicht, wenn der Prozess ausgelagert (Paging) wird. Stellen Sie für solche Systeme sicher, dass Sie Paging-Bereich in der Größe des gesamten gemeinsamen Speicherbereichs bereitstellen.

Bei der Mehrzahl der Konfigurationsparameter wird der entsprechende Speicher erst zugeordnet, wenn er angefordert wird. Diese Parameter legen die maximale Größe eines bestimmten Zwischenspeicherbereichs fest. In den folgenden Fällen wird jedoch die volle Größe des Speichers zugeordnet, die durch den Parameter angegeben wird:

- Maximaler Speicher für Sperrenliste (*locklist*)
- Zwischenspeichergröße für Anwendungsunterstützungsebene (*aslheapsz*)
- Anzahl der FCM-Puffer (*fcm_num_buffers*)

Anmerkung: Zur Änderung der Größe eines Pufferpools verwenden Sie die DDL-Anweisung ALTER BUFFERPOOL.

Parameter mit Auswirkung auf Speichernutzung durch Anwendungsgruppen

Die Parameter, die sich auf die Nutzung von Speicher durch Anwendungsgruppen auswirken, gelten nur für partitionierte Datenbanken, für Datenbanken mit aktivierter partitionsinterner Parallelverarbeitung sowie für Datenbanken, für die der Verbindungskonzentrator aktiviert ist. Die folgenden Parameter legen fest, wie Anwendungen in Anwendungsgruppen ihren gemeinsamen Speicher verwenden:

- Der Parameter *appgroup_mem_sz* gibt die Größe des gemeinsamen Speichers für die Anwendungsgruppe an.

  Eine zu hohe Einstellung des Parameters *appgroup_mem_sz* hat einen negativen Effekt. Da alle Anwendungen in der Anwendungsgruppe die Caches im Zwischenspeicher der Anwendungsgruppe gemeinsam verwenden, erhöhen zu viele Anwendungen die Konkurrenzsituation bei der Cachenutzung. Wenn jede Anwendungsgruppe andererseits zu wenig Anwendungen enthält, ist die Wirkung des Cache ebenfalls begrenzt.
- Der Parameter *groupheap_ratio* gibt den zulässigen Prozentsatz an Speicher für den gemeinsamen Speicher an.

  Eine zu niedrige Einstellung des Parameters *groupheap_ratio* beschränkt die Größe von Caches. Eine zu hohe Einstellung von *groupheap_ratio* bewirkt, dass der Steuerungszwischenspeicher für Anwendungen zu klein wird und verursacht eventuell den SQL-Fehler `SQL0973`, der Sie darauf hinweist, dass zur Laufzeit zu wenig Speicher für den Zwischenspeicher zur Anwendungssteuerung verfügbar ist.
- Der Parameter *app_ctl_heap_sz* gibt die Größe des Steuerungszwischenspeichers für jede Anwendung in der Gruppe an.

Übernehmen Sie die Standardwerte für diese Parameter, wenn Sie Ihren Datenbankserver konfigurieren. Passen Sie die Einstellungen nur an, wenn die Leistung beeinträchtigt ist. Definieren Sie zum Beispiel den Parameter *appgroup_mem_sz*, um die Anzahl von Anwendungen in jeder Anwendungsgruppe zu steuern. Als Faustregel ist zu beachten, dass 10 zu wenige und 100 zu viele sind. Der Standardwert ist wahrscheinlich angemessen. Führen Sie anschließend eine durchschnittliche Auslastung aus und verwenden Sie die Diagnosezentrale in der Steuerzentrale oder den Systemmonitor, um Informationen zu den Trefferquoten für den Katalogcache, den Paketcache und den gemeinsamen Arbeitsbereich zu erfassen.

- Wenn zahlreiche Fehler `sql0973` auftreten, ist die Einstellung für *groupheap_ratio* zu hoch.
- Wenn der Speichermonitor zeigt, dass die durchschnittliche und die maximale Nutzung des Zwischenspeichers für die Anwendungssteuerung weit unter dem Wert `app_ctl_heap_sz * (100 - groupheap_ratio)/ 100` liegt, verringern Sie den Wert des Konfigurationsparameters *app_ctl_heap_sz*.
- Wenn die Werte für die Cachenutzung darauf hinweisen, dass die Caches ihre Grenze erreichen, erhöhen Sie den Wert des Konfigurationsparameters *group_heap_ratio* oder verringern die Anzahl von Anwendungen in der Anwendungsgruppe.

Hinweise:

- Vergleichstests liefern die besten Informationen zur Einstellung geeigneter Werte für Speicherparameter. Bei der Durchführung von Vergleichstests werden repräsentative SQL-Anweisungen und Extremfall-SQL-Anweisungen für den Server ausgeführt und die Werte der Parameter modifiziert, bis der Punkt gefunden wird, an dem die Leistung wieder sinkt. Wenn die Leistung gegen die Parameterwerte in einem Diagramm aufgezeichnet würde, zeigt die

Stelle, an dem die Kurve nicht weiter steigt bzw. wieder zu sinken beginnt, den Punkt an, an dem eine weitere Speicherzuordnung keinen weiteren Vorteil für die Anwendung bringt und daher nur zu einer Verschwendung von Speicher führen würde.

- Die oberen Grenzen der Speicherzuordnung einiger Parameter können über den Speicherkapazitäten der vorhandenen Hardware und des Betriebssystems liegen. Diese Grenzen tragen steigenden Anforderungen in der Zukunft Rechnung.
- Gültige Wertebereiche der Parameter finden Sie in den detaillierten Informationen zu den einzelnen Parametern.

Zugehörige Konzepte:

- „Organisation der Speichernutzung" auf Seite 247
- „Gemeinsam benutzter Speicher des Datenbankmanagers" auf Seite 250
- „Globaler Speicher und zugehörige Steuerparameter" auf Seite 253

Zugehörige Referenzen:

- „app_ctl_heap_sz - Zwischenspeichergröße für Anwendungssteuerung" auf Seite 406
- „fcm_num_buffers - Anzahl FCM-Puffer" auf Seite 517
- „sheapthres - Schwellenwert für Sortierspeicher" auf Seite 416
- „aslheapsz - Zwischenspeichergröße für Anwendungsunterstützungsebene" auf Seite 421
- „pckcachesz - Größe des Paketcache" auf Seite 403
- „locklist - Maximaler Speicher für Sperrenliste" auf Seite 399
- „dbheap - Zwischenspeicher für Datenbank" auf Seite 399
- „util_heap_sz - Zwischenspeichergröße für Dienstprogramme" auf Seite 406
- „catalogcache_sz - Katalogcachegröße" auf Seite 396
- „appgroup_mem_sz - Maximale Speichergröße für Anwendungsgruppe" auf Seite 409
- „groupheap_ratio - Für Anwendungsgruppenzwischenspeicher vorgesehener Speicher in Prozent" auf Seite 410

Pufferpools

Pufferpools bilden eine Speicherkomponente von entscheidender Bedeutung. Dieser Abschnitt beschreibt Pufferpools und bietet Informationen zu ihrer Verwaltung im Hinblick auf optimale Leistung.

Pufferpoolverwaltung

Ein Pufferpool ist ein Speicher, der beim Lesen von Daten vom Datenträger oder beim Ändern von Daten zum Zwischenspeichern von Tabellen- und Indexdatenseiten verwendet wird. Der Pufferpool verbessert die Leistung des Datenbanksystems, indem er die Möglichkeit schafft, auf Daten im Hauptspeicher anstatt auf der Platte zuzugreifen. Da der Zugriff auf den Hauptspeicher wesentlich schneller als der Plattenzugriff arbeitet, ist die Leistung umso höher, je seltener der Datenbankmanager Daten von der Platte lesen bzw. auf die Platte schreiben muss. Aufgrund der Tatsache, dass die meisten Bearbeitungsschritte für Daten in Pufferpools erfolgen, ist die Konfiguration von Pufferpools der wichtigste Einzelbereich bei der Optimierung. Nur LOB-Daten und Langfelddaten werden nicht in einem Pufferpool bearbeitet.

Wenn eine Anwendung auf eine Zeile einer Tabelle zum ersten Mal zugreift, liest der Datenbankmanager die Seite, die die Zeile enthält, in den Pufferpool ein. Wenn in der Folge eine Anwendung Daten anfordert, sucht sie der Datenbankmanager im Pufferpool. Befinden sich die angeforderten Daten im Pufferpool, können sie ohne Plattenzugriff abgerufen werden, was eine höhere Geschwindigkeit ermöglicht.

Der Speicher für den Pufferpool wird zugeordnet, wenn eine Datenbank aktiviert wird oder wenn die erste Anwendung die Verbindung zur Datenbank herstellt. Pufferpools können auch erstellt, gelöscht oder in der Größe geändert werden, während der Datenbankmanager aktiv ist. Wenn Sie das Schlüsselwort IMMEDIATE in der Anweisung ALTER BUFFERPOOL zur Vergrößerung des Pufferpools verwenden, wird der Speicher zugeordnet, sobald Sie den Befehl eingegeben haben, sofern der Speicher verfügbar ist. Wenn der Speicher nicht verfügbar ist, findet die Änderung statt, wenn alle Anwendungen ihre Verbindungen getrennt haben und die Datenbank erneut aktiviert wird. Wenn Sie den Pufferpool verkleinern, wird der Speicher bei der Festschreibung (COMMIT) freigegeben. Wenn alle Anwendungen getrennt sind, wird der Pufferpoolspeicher freigegeben.

Anmerkung: Zur Verringerung der Notwendigkeit, den Wert des Datenbankkonfigurationsparameters *dbheap* zu erhöhen, wenn der Pufferpool vergrößert wird, wird beinahe der gesamte Pufferpoolspeicher, der auch Seitendeskriptoren, Pufferpooldeskriptoren und die Hashtabellen umfasst, dem gemeinsamen Datenbankspeicher entnommen und automatisch in der Größe angepasst.

Um sicherzustellen, dass in allen Situationen ein geeigneter Pufferpool verfügbar ist, erstellt DB2® kleine Pufferpools, d. h. je einen mit der Seitengröße 4 KB, 8 KB, 16 KB und 32 KB. Die Größe der Pufferpools beträgt 16 Seiten. Diese Pufferpools sind für den Benutzer verdeckt. Sie treten weder in den Systemkatalogen noch in den Pufferpoolsystemdateien auf. Diese Pufferpools können von Ihnen auch nicht direkt verwendet oder geändert werden, sondern sie werden von DB2 in den folgenden Situationen genutzt:

- Wenn ein Pufferpool der erforderlichen Seitengröße nicht aktiv ist, weil nicht genügend Speicher zu seiner Erstellung verfügbar war, nachdem eine Anweisung CREATE BUFFERPOOL mit dem Schlüsselwort IMMEDIATE ausgeführt wurde.

  Eine Nachricht wird in das Benachrichtigungsprotokoll für die Systemverwaltung geschrieben. Falls erforderlich, werden Tabellenbereiche einem verdeckten Pufferpool zugeordnet. Die Leistung könnte sich beträchtlich verringern.
- Wenn die normalen Pufferpools bei der Herstellung einer Datenbankverbindung nicht aktiviert werden können.

  Dieses Problem hat wahrscheinlich eine ernsthafte Ursache, wie zum Beispiel, dass kein Speicher mehr verfügbar ist. Obwohl DB2 aufgrund der verdeckten Pufferpools die volle Funktionsfähigkeit bewahrt, wird sich die Leistung erheblich verschlechtern. Sie sollten dieses Problem unverzüglich untersuchen. Wenn dieser Fall eintritt, empfangen Sie eine Warnung und eine Nachricht wird in das Benachrichtigungsprotokoll für die Systemverwaltung geschrieben.

Seiten verbleiben im Pufferpool, bis die Datenbank gestoppt wird oder der von einer Seite belegte Speicherbereich im Pufferpool für eine andere Seite benötigt wird. Die folgenden Bedingungen bestimmen, welche Seite entfernt wird, um ein andere Seite einzulesen:

- Der letzte Verweis auf die Seite

- Die Wahrscheinlichkeit, dass der letzte Agent, der die Seite gelesen hat, erneut auf die Seite zugreift
- Der Typ der Daten auf der Seite
- Der Änderungsstatus der Seite, d. h. ob sie im Speicher geändert, jedoch noch nicht auf Platte geschrieben wurde (geänderte Seiten werden immer auf der Festplatte gespeichert, bevor sie im Pufferpool überschrieben werden)

Damit erneut auf Seiten im Speicher zugegriffen werden kann, werden geänderte Seiten nicht aus dem Pufferpool entfernt, nachdem sie auf die Festplatte geschrieben wurden, sofern der von ihnen belegte Speicherplatz nicht benötigt wird.

Wenn Sie einen Pufferpool erstellen, beträgt die Standardseitengröße 4 KB, jedoch können Sie auch eine Seitengröße von 4 KB, 8 KB, 16 KB oder 32 KB angeben. Da Seiten in einen Pufferpool nur eingelesen werden können, wenn die Seitengröße des Tabellenbereichs mit der Seitengröße des Pufferpools übereinstimmt, sollte die Seitengröße Ihrer Tabellenbereiche auch die Seitengröße sein, die Sie für Pufferpools angeben. Nach der Erstellung eines Pufferpools können Sie seine Seitengröße nicht mehr ändern. Sie müssen einen neuen Pufferpool mit einer anderen Seitengröße erstellen.

Anmerkung: Auf 32-Bit-Plattformen unter Windows® NT bzw. unter Windows 2000 Advanced Server oder Data Center können Sie große Pufferpools erstellen, wenn Sie Address Windowing Extensions (AWE) aktiviert haben.

Zugehörige Konzepte:

- „Organisation der Speichernutzung" auf Seite 247
- „Sekundäre Pufferpools im erweiterten Speicher auf 32-Bit-Plattformen" auf Seite 259
- „Pufferpoolverwaltung von Datenseiten" auf Seite 261
- „Illustration der Datenseitenverwaltung in Pufferpools" auf Seite 264
- „Verwaltung mehrerer Datenbankpufferpools" auf Seite 265

Sekundäre Pufferpools im erweiterten Speicher auf 32-Bit-Plattformen

Auf 64-Bit-Plattformen kann auf große adressierbare virtuelle Speicher in üblicher Weise ohne spezielle Techniken zugegriffen werden. Auf 32-Bit-Plattformen ist der adressierbare virtuelle Speicher in der Regel auf eine Größe zwischen 2 GB und 4 GB begrenzt. Wenn Ihre 32-Bit-Maschine über mehr adressierbaren Realspeicher als diese Maximalgröße verfügt, können Sie den zusätzlichen adressierbaren Realspeicher oberhalb des adressierbaren virtuellen Speichers als *Erweiterungsspeichercache* konfigurieren. Jeder der definierten Pufferpools kann einen Erweiterungsspeichercache zur Verbesserung der Leistung nutzen. Sie definieren den Erweiterungsspeichercache als eine Anzahl von Speichersegmenten. Wenn Sie einen Teil des adressierbaren Realspeichers als Erweiterungsspeichercache definieren, kann dieser Speicher nicht länger für andere Zwecke genutzt werden, wie zum Beispiel als JFS-Cache oder als privater Prozessadressraum. Wenn Sie adressierbaren Realspeicher einem Erweiterungsspeichercache zuordnen, kann es zu erhöhtem System-Paging kommen.

Die Pufferpools führen das Caching der ersten Ebene durch, während ein vorhandener Erweiterungsspeichercache von den Pufferpools als Cache der sekundären Ebene verwendet wird. Im Idealfall enthalten die Pufferpools die Daten, auf die

am häufigsten zugegriffen wird, während der Erweiterungsspeichercache Daten enthält, auf die nicht so häufig zugegriffen wird.

Anmerkung: Sie können AWE-Pufferpools (AWE - Address Windowing Extensions) unter Windows® 2000 mit Hilfe der Registriervariablen DB2_AWE zuordnen. Windows AWE ist eine Gruppe von Speicherverwaltungserweiterungen, die es Anwendungen ermöglichen, Speicherbereiche oberhalb bestimmter Grenzen zu nutzen, die von dem Prozessmodell der jeweiligen Anwendung abhängig sind. Informationen finden Sie in der Dokumentation zu Ihrem Windows-System. Beachten Sie jedoch, dass Sie bei Verwendung des Speichers zu diesem Zweck den Erweiterungsspeichercache nicht nutzen können.

Die folgenden Datenbankkonfigurationsparameter beeinflussen die Anzahl und die Größe der für den Erweiterungsspeichercache verfügbaren Speichersegmente:

- *num_estore_segs* definiert die Anzahl der Segmente für den Erweiterungsspeichercache. Der Standardwert für diesen Konfigurationsparameter ist null, d. h., es ist kein Erweiterungsspeichercache vorhanden.
- *estore_seg_sz* definiert die Größe der einzelnen Segmente des Erweiterungsspeichers. Diese Größe wird durch die Plattform festgelegt, auf der der Erweiterungsspeichercache verwendet wird.

Da der Erweiterungsspeichercache eine Erweiterung eines Pufferpools darstellt, muss er immer einem oder mehreren bestimmten Pufferpools zugeordnet werden. Das bedeutet, dass Sie definieren müssen, welche Pufferpools einen Cache, wenn er erstellt ist, nutzen können. Die Anweisungen CREATE und ALTER BUFFERPOOL haben die Attribute NOT EXTENDED STORAGE und EXTENDED STORAGE, die die Verwendung des Cache steuern. Standardmäßig verwenden weder der Pufferpool IBMDEFAULTBP noch irgendein neu erstellter Pufferpool den erweiterten Speicher.

Anmerkung: Wenn Sie Pufferpools mit verschiedenen definierten Seitengrößen verwenden, kann jeder dieser Pufferpools zur Verwendung des erweiterten Speichers definiert werden. Die für die Erweiterungsspeicherunterstützung verwendete Seitengröße ist die größte definierte Seitengröße.

Obwohl der Datenbankmanager Daten, die sich im Erweiterungsspeichercache befinden, nicht direkt bearbeiten kann, kann er Daten aus dem Erweiterungsspeichercache viel schneller als vom Plattenspeicher in den Pufferpool übertragen.

Wenn eine Zeile von Daten einer Seite im Erweiterungsspeichercache benötigt wird, wird die gesamte Seite in den entsprechenden Pufferpool eingelesen.

Ein Pufferpool und der definierte zugeordnete Erweiterungsspeichercache werden zugeordnet, wenn eine Datenbank aktiviert oder die erste Verbindung zu ihr hergestellt wird.

Zugehörige Konzepte:

- „Pufferpoolverwaltung" auf Seite 257
- „Speicherverwaltung" auf Seite 38

Zugehörige Referenzen:

- „estore_seg_sz - Segmentgröße für erweiterten Speicher" auf Seite 438
- „num_estore_segs - Segmentanzahl für erweiterten Speicher" auf Seite 438

Pufferpoolverwaltung von Datenseiten

Seiten im Puffer können entweder *im Gebrauch* oder nicht, und sie können *benutzt* (geändert) oder *sauber* (ungeändert) sein:

- Seiten, die im Gebrauch sind, werden momentan gelesen oder aktualisiert. Während eine Seite durch einen Agenten im Gebrauch ist, kann sie von anderen Agenten oder Vorablesefunktionen in der Datenbank gelesen, jedoch nicht aktualisiert werden.
- Benutzte Seiten enthalten Daten, die geändert, jedoch noch nicht auf die Platte geschrieben wurden.
- Wenn eine Seite auf die Platte geschrieben wurde, ist sie sauber, verbleibt jedoch im Pufferpool, bis Platz für neue Seiten benötigt wird. Saubere Seiten können auch in einen zugeordneten Erweiterungsspeichercache verlegt werden, wenn einer definiert ist.

Agenten für Seitenlöschfunktionen

In einem entsprechend optimierten System schreiben meist Agenten für Seitenlöschfunktionen benutzte Seiten auf die Platte. Agenten für Seitenlöschfunktionen führen E/A-Operationen als Hintergrundprozesse aus und ermöglichen Anwendungen eine schnellere Ausführung, weil ihre Agenten die tatsächlichen Transaktionen ausführen können. Agenten für Seitenlöschfunktionen können auch als *asynchrone Seitenlöschfunktionen* oder *asynchrone Pufferschreibfunktionen* bezeichnet werden, da sie nicht mit der Arbeit anderer Agenten koordiniert werden und nur bei Bedarf aktiv werden.

Zur Verbesserung der Leistung für aktualisierungsintensive Auslastungen kann es nützlich sein, mehr Agenten für Seitenlöschfunktionen zu konfigurieren. Die Leistung kann sich erhöhen, wenn mehr Seitenlöschagenten verfügbar sind, um benutzte Seiten auf die Festplatte zu schreiben. Dies gilt insbesondere, wenn Momentaufnahmen offenbaren, dass eine erhebliche Anzahl synchroner Schreiboperationen für Daten- oder Indexseiten im Verhältnis zur Anzahl asynchroner Schreiboperationen für Daten- oder Indexseiten stattfindet.

Seitenbereinigung und schnelle Wiederherstellung

Wenn mehr Seiten auf die Festplatte geschrieben wurden, ist die Wiederherstellung der Datenbank nach einem Systemabsturz schneller, weil der Datenbankmanager größere Teile des Pufferpools von der Festplatte rekonstruieren kann, anstatt Transaktionen aus den Datenbankprotokolldateien erneut durchzuführen.

Die Größe des Protokolls, das während der Wiederherstellung gelesen werden muss, errechnet sich aus der Differenz der Positionen der folgenden Datensätze im Protokoll:

- Der zuletzt geschriebene Protokollsatz
- Der Protokollsatz, der die älteste Änderung an Daten im Pufferpool beschreibt

Die Standardfunktionsweise der Seitenlöschfunktionen sieht vor, dass eine Bereinigung von Seiten durchgeführt wird, wenn die Größe des Protokolls, das während einer Wiederherstellung wieder angewendet werden müsste, den folgenden Maximalwert überschreitet:

```
logfilsiz * softmax
```

Dabei gilt Folgendes:

- *logfilsiz* ist die Größe der Protokolldateien
- *softmax* ist der Prozentsatz der Protokolldateien, die nach einem Datenbankabsturz wiederherzustellen sind. Wenn der Parameter *softmax* zum Beispiel den Wert 250 hat, enthalten 2,5 Protokolldateien die Änderungen, die wiederhergestellt werden müssen, wenn ein Systemabsturz auftritt.

Ermitteln Sie zur Minimierung der Zeit, die bei einer Wiederherstellung auf das Lesen der Protokolle verwendet wird, mit Hilfe des Datenbanksystemmonitors die Häufigkeit, mit der Seitenlöschfunktionen aktiv werden. Das Monitorelement *pool_lsn_gap_clns (für Pufferpoolprotokollbereich ausgelöste Seitenlöschfunktionen)* des Systemmonitors stellt diese Information bereit, wenn Sie nicht die proaktive Seitenbereinigung für Ihre Datenbank aktiviert haben. Wenn Sie diese alternative Seitenbereinigungsfunktion aktiviert haben, sollte diese Situation nicht eintreten und das Monitorelement *pool_lsn_gap_clns* zeigt immer den Wert 0.

Mit Hilfe des Monitorelements *log_held_by_dirty_pages* können Sie bestimmen, ob die Seitenlöschfunktionen nicht ausreichend Seiten bereinigen, um die vom Benutzer festgelegten Bindungen für die Wiederherstellung zu erfüllen. Wenn das Monitorelement *log_held_by_dirty_pages* beständig einen erheblich größeren Wert als *logfilsiz* * *softmax* zeigt, sind entweder mehr Seitenlöschfunktionen erforderlich oder der Wert des Parameters *softmax* muss angepasst werden.

Zugehörige Konzepte:

- „Illustration der Datenseitenverwaltung in Pufferpools" auf Seite 264

Zugehörige Referenzen:

- „Leistungsvariablen" auf Seite 586

Proaktive Seitenbereinigung

Seit Version 8.1.4 steht eine alternative Methode zur Konfiguration der Seitenbereinigung in Ihrem System zur Verfügung. Diese alternative Methode unterscheidet sich insofern von der Standardfunktionsweise, als dass die Seitenlöschfunktionen sich proaktiver bei der Auswahl der benutzten (geänderten) Seiten verhalten, die zu einem gegebenen Zeitpunkt ausgelesen und auf die Platte geschrieben werden. Diese neue Methode der Seitenbereinigung unterscheidet sich von der Standardseitenbereinigung in zwei wesentlichen Punkten:

1. Die Seitenlöschfunktionen werden nicht durch den Konfigurationsparameter 'chngpgs_thresh' beeinflusst.

   Bei dieser alternativen Methode der Seitenbereinigung reagieren Seitenlöschfunktionen nicht mehr auf den Wert des Konfigurationsparameters 'chngpgs_thresh'. Anstatt zu versuchen, einen Prozentsatz des Pufferpools sauber zu halten, stellt die alternative Methode der Seitenbereinigung einen Mechanismus bereit, durch den Agenten über die Position gut geeigneter Seiten, die gerade auf die Platte geschrieben wurden, informiert werden, so dass Agenten den Pufferpool nicht durchsuchen müssen, um geeignete Seiten zu finden. Wenn die Anzahl gut geeigneter Seiten unter einen akzeptablen Wert fällt, werden die Seitenlöschfunktionen ausgelöst, die daraufhin den gesamten Pufferpool durchsuchen und potenziell geeignete Seiten auf die Platte schreiben und die Agenten über die Positionen dieser Seiten informieren.

2. Seitenfunktionen reagieren nicht mehr auf Auslöser bei LSN-Abstimmungsverlusten (LSN - Protokollfolgenummer), die von der Protokollfunktion abgesetzt werden.

   Wenn die Größe des Protokollspeichers, der den Protokollsatz enthält, der die älteste Seite im Pufferpool aktualisiert hat, und die aktuelle Protokollposition den vom Parameter 'softmax' zugelassenen Wert überschreiten, wird diese Situation als LSN-Abstimmungsverlust (LSNGAP-Bedingung) der Datenbank bezeichnet. Wenn die Protokollfunktion unter der Standardmethode der Seitenbereinigung erkennt, dass ein LSN-Abstimmungsverlust aufgetreten ist, startet sie die Seitenlöschfunktionen, um alle Seiten, die zu dieser Situation beitragen, auf die Platte zu schreiben. Das heißt, sie schreibt die Seiten, die älter sind, als durch den Parameter 'softmax' zugelassen wird, auf die Platte. Seitenlöschfunktionen sind über einen gewissen Zeitraum, in dem kein LSN-Abstimmungsverlust auftritt, nicht aktiv. Wenn dann ein LSN-Abstimmungsverlust auftritt, werden die Seitenlöschfunktionen aktiviert und schreiben eine große Anzahl von Seiten auf die Platte, bevor sie wieder inaktiv werden. Dies kann zu einer Sättigung des E/A-Subsystems führen, die wiederum andere Agenten beeinträchtigen kann, die gerade Seiten lesen oder schreiben. Darüber hinaus ist es möglich, das bis zu dem Zeitpunkt, zu dem ein LSN-Abstimmungsverlust ausgelöst wird, die Seitenlöschfunktionen die Seiten schon nicht mehr schnell genug bereinigen können und DB2® am Ende keinen Protokollspeicherbereich mehr zur Verfügung hat.

   Die alternative Methode der Seitenbereinigung moduliert diese Funktionsweise, indem sie die gleiche Anzahl von Schreiboperationen über einen größeren Zeitraum verteilt. Die Löschfunktionen führen dies aus, indem sie proaktiv nicht nur Seiten bereinigen, die sich zurzeit in einer Situation mit LSN-Abstimmungsverlust befinden, sondern auch die Seiten, die in Anbetracht des aktuellen Aktivitätsvolumens bald in eine solche Situation kommen werden.

Zur Verwendung der neuen Methode der Seitenbereinigung setzen Sie die Registriervariable DB2_USE_ALTERNATE_PAGE_CLEANING auf den Wert ON.

Zugehörige Konzepte:

- „Pufferpoolverwaltung von Datenseiten" auf Seite 261

Illustration der Datenseitenverwaltung in Pufferpools

Die folgende Abbildung veranschaulicht, wie die Arbeit zur Verwaltung des Pufferpools zwischen den Agenten für Seitenlöschfunktionen und den Datenbankagenten aufgeteilt werden kann, im Vergleich zu der Situation, in der die Datenbankagenten die gesamte Ein-/Ausgabe selbst durchführen.

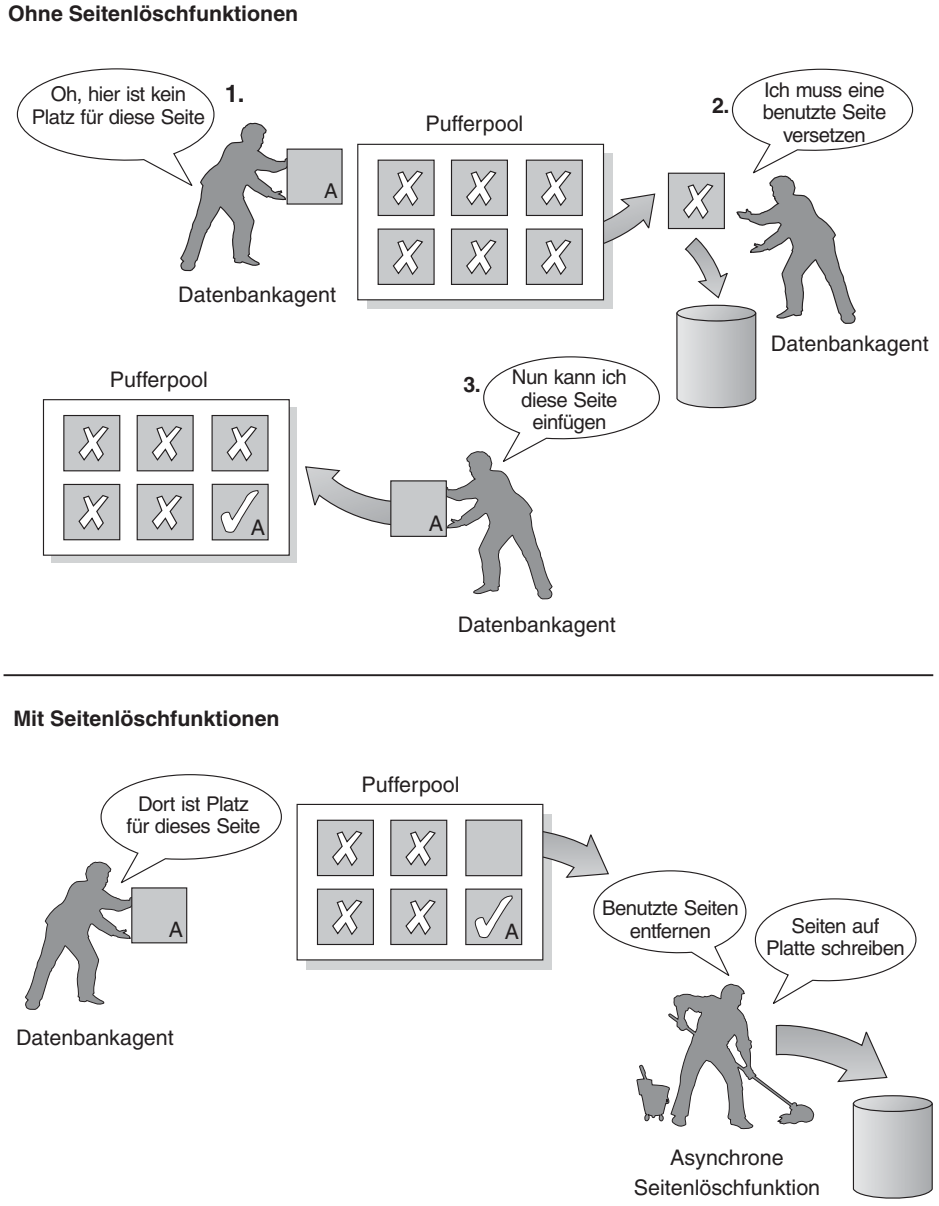


Abbildung 23. Asynchrone Seitenlöschfunktion. Geänderte Seiten werden auf Platte geschrieben.

Zugehörige Konzepte:

- „Pufferpoolverwaltung" auf Seite 257
- „Pufferpoolverwaltung von Datenseiten" auf Seite 261

Verwaltung mehrerer Datenbankpufferpools

Obwohl jede Datenbank mindestens einen Pufferpool erfordert, können Sie für eine Datenbank, die Tabellenbereiche mit mehr als einer Seitengröße hat, mehrere Pufferpools erstellen, jeden in einer anderen Größe oder mit einer anderen Seitengröße. Jeder Pufferpool hat eine Mindestgröße, die von der Plattform abhängig ist.

Eine neue Datenbank besitzt einen Standardpufferpool namens IBMDEFAULTBP mit einer Größe, die von der Plattform festgelegt wird, und der Standardseitengröße von 4 KB. Wenn Sie einen Tabellenbereich mit 4-KB-Seiten erstellen und keinem bestimmten Pufferpool zuordnen, wird der Tabellenbereich dem Standardpufferpool zugeordnet. Sie können die Größe des Standardpufferpools und seine Attribute ändern, jedoch können Sie ihn nicht löschen.

Anmerkung: Während des normalen Datenbankmanagerbetriebs können Sie die Größe eines Pufferpools mit Hilfe des Befehls ALTER BUFFERPOOL ändern.

Seitengrößen für Pufferpools

Nach der Erstellung oder Migration einer Datenbank können Sie weitere Pufferpools erstellen. Zum Beispiel könnten Sie bei der Planung Ihrer Datenbank feststellen, dass sich Seitengrößen von 8 KB am besten für Tabellen eignen. In diesem Fall sollten Sie einen Pufferpool mit 8-KB-Seiten sowie einen oder mehrere Tabellenbereiche mit derselben Seitengröße erstellen. Mit der Anweisung ALTER TABLESPACE können Sie einen Tabellenbereich keinem Pufferpool zuordnen, der eine andere Seitengröße verwendet.

Anmerkung: Wenn Sie einen Tabellenbereich mit einer Seitengröße über 4 KB (z. B. 8 KB, 16 KB oder 32 KB) erstellen, müssen Sie ihn einem Pufferpool zuordnen, der dieselbe Seitengröße verwendet. Wenn dieser Pufferpool momentan nicht aktiv ist, versucht DB2®, den Tabellenbereich vorübergehend einem anderen aktiven Pufferpool mit der gleichen Seitengröße zuzuordnen, falls einer vorhanden ist, bzw. einem der „verdeckten" Pufferpools, die DB2 standardmäßig erstellt, wenn der erste Client eine Verbindung zur Datenbank herstellt. Ist der ursprünglich angegebene Pufferpool beim nächsten Aktivieren der Datenbank aktiv, ordnet DB2 den Tabellenbereich diesem Pufferpool zu.

Wenn Sie einen Pufferpool erstellen, geben Sie die Größe des Pufferpools als erforderlichen Parameter der DDL-Anweisung CREATE BUFFERPOOL an. Wenn Sie den Pufferpool später vergrößern oder verkleinern, verwenden Sie die DDL-Anweisung ALTER BUFFERPOOL.

In einer partitionierten Datenbankumgebung besitzt jeder Pufferpool für eine Datenbank in allen Datenbankpartitionen die gleiche Standarddefinition, sofern dies nicht in der Anweisung CREATE BUFFERPOOL anders angegeben oder die Größe des Pufferpools für eine bestimmte Datenbankpartition mit der Anweisung ALTER BUFFERPOOL geändert wurde.

Vorteile großer Pufferpools

Große Pufferpools bieten die folgenden Vorteile:
- Sie ermöglichen, dass häufig angeforderte Datenseiten im Pufferpool behalten werden können. Dadurch kann schneller auf sie zugegriffen werden. Durch weniger E/A-Operationen können E/A-Konkurrenzsituationen besser vermieden werden, so dass die Antwortzeiten verbessert und die für E/A-Operationen benötigten Prozessorressourcen reduziert werden.
- Sie bieten die Möglichkeit, höhere Transaktionsgeschwindigkeiten mit derselben Antwortzeit zu erzielen.
- Sie vermeiden E/A-Konkurrenzsituationen für häufig verwendete Plattenspeichereinheiten zum Beispiel für Katalogtabellen, häufig verwendete Benutzertabellen und Indizes. Auch Sortierungen durch Abfragen profitieren von verminderten E/A-Konkurrenzsituationen auf Plattenspeichereinheiten, die die temporären Tabellenbereiche enthalten.

Vorteile vieler Pufferpools

Wenn eine der folgenden Bedingungen auf Ihr System zutrifft, sollten Sie nur einen einzigen Pufferpool verwenden:
- Der gesamte Pufferspeicherbereich beträgt weniger als 10 000 4-KB-Seiten.
- Es stehen keine Fachleute mit Anwendungskenntnissen für die spezialisierte Optimierung zur Verfügung.
- Sie arbeiten auf einem Testsystem.

In allen anderen Fällen sollten Sie die Verwendung mehrerer Pufferpools aus folgenden Gründen in Betracht ziehen:
- Tabellenbereiche für temporäre Tabellen können einem getrennten Pufferpool zugeordnet werden, um eine bessere Leistung für Abfragen zu erzielen, die temporären Speicherbereich benötigen, insbesondere Abfragen mit zahlreichen Sortierungen.
- Wenn auf Daten wiederholt und schnell von vielen kurzen Anwendungen mit Aktualisierungstransaktionen zugegriffen werden muss, ziehen Sie in Betracht, den Tabellenbereich, der die Daten enthält, einem getrennten Pufferpool zuzuordnen. Wenn dieser Pufferpool eine geeignete Größe hat, ist die Wahrscheinlichkeit höher, dass die Seiten im Pufferpool gefunden werden. Dies trägt zur Verkürzung der Antwortzeiten und zur Reduzierung der Transaktionskosten bei.
- Sie können Daten in getrennten Pufferpools isolieren, um eine bevorzugte Verarbeitung bestimmter Anwendungen, Daten und Indizes zu erreichen. Zum Beispiel könnte es sinnvoll sein, Tabellen und Indizes, die häufig aktualisiert werden, in einen Pufferpool einzulesen, der von anderen Tabellen und Indizes, die zwar häufig abgefragt, aber nicht häufig aktualisiert werden, getrennt ist. Durch diese Änderung werden die Auswirkungen verringert, die häufige Aktualisierungen an der ersten Gruppe von Tabellen auf häufige Abfragen auf die zweite Gruppe von Tabellen haben.
- Sie können kleinere Pufferpools für die Daten verwenden, auf die Anwendungen zugreifen, die sehr selten ausgeführt werden, insbesondere wenn eine solche Anwendung einen sehr wahlfreien Zugriff auf eine sehr umfangreiche Tabelle benötigt. In diesem Fall brauchen die Daten nicht länger als für eine einzige Abfrage im Pufferpool behalten zu werden. Es ist günstiger, einen kleinen Pufferpool für diese Daten zu verwenden und den übrigen Speicher für andere Zwecke freizugeben, wie zum Beispiel für andere Pufferpools.

- Nach der Trennung der verschiedenen Aktivitäten und Daten in verschiedene Pufferpools können gute und relativ unaufwendige Daten zur Leistungsdiagnose aus den Statistiken und aus Tracefunktionen für Benutzeraktivitäten gewonnen werden.

Zuordnung von Pufferpoolspeicher beim Start

Wenn Sie einen Pufferpool mit dem Befehl CREATE BUFFERPOOL erstellen oder Pufferpools mit dem Befehl ALTER BUFFERPOOL ändern, muss der gesamte Speicherbereich, der von allen Pufferpools benötigt wird, dem Datenbankmanager zur Verfügung stehen, damit alle Pufferpools beim Starten der Datenbank zugeordnet werden können. Wenn Sie Pufferpools erstellen oder modifizieren, während der Datenbankmanager online ist, sollte zusätzlicher Speicherplatz im globalen Datenbankspeicher verfügbar sein. Wenn Sie das Schlüsselwort IMMEDIATE bei der Erstellung eines neuen Pufferpools bzw. bei der Vergrößerung eines vorhandenen Pufferpools angeben und der erforderliche Speicherplatz nicht verfügbar ist, führt der Datenbankmanager die Änderung bei der nächsten Aktivierung der Datenbank durch. Auf 32-Bit-Plattformen muss der Speicherplatz verfügbar sein und kann im globalen Datenbankspeicher reserviert werden, wie in den Detailinformationen zum Datenbankkonfigurationsparameter *database_memory* beschrieben wird.

Falls dieser Speicher beim Starten einer Datenbank nicht verfügbar ist, versucht der Datenbankmanager jeweils einen der mit unterschiedlichen Seitengrößen definierten Pufferpools für jede Seitengröße zu starten. Allerdings werden die Pufferpools nur mit einer minimalen Größe von jeweils 16 Seiten gestartet. Zur Angabe einer anderen minimalen Pufferpoolgröße verwenden Sie die Registriervariable DB2_OVERRIDE_BPF. Immer wenn ein Pufferpool beim Starten nicht zugeordnet werden kann, wird eine Warnung SQL1478W (SQLSTATE 01626) zurückgegeben. Die Datenbank setzt den Betrieb in diesem Status fort, bis ihre Konfiguration geändert wird und die Datenbank vollständig neu gestartet werden kann.

Der Datenbankmanager wird mit den minimal dimensionierten Werten nur deshalb gestartet, um Ihnen die Möglichkeit zu geben, eine Verbindung zur Datenbank herzustellen und die Pufferpoolgrößen anders zu konfigurieren bzw. andere wichtige Maßnahmen durchzuführen. Starten Sie die Datenbank neu, sobald Sie diese Maßnahmen durchgeführt haben. Lassen Sie die Datenbank nicht über längere Zeit in einem solchen Status arbeiten.

Zugehörige Konzepte:

- „Pufferpoolverwaltung" auf Seite 257
- „Sekundäre Pufferpools im erweiterten Speicher auf 32-Bit-Plattformen" auf Seite 259

Konzepte des Vorablesens

Das Vorablesen von Daten in die Pufferpools verbessert in der Regel die Leistung, da die Anzahl von Datenträgerzugriffen verringert und häufig genutzte Daten im Speicher behalten werden.

Vorablesen von Daten in den Pufferpool

Vorablesen von Seiten bedeutet, dass eine oder mehrere Seiten von der Platte in der Erwartung abgerufen werden, dass sie von einer Anwendung angefordert werden. Durch das Vorablesen von Index- und Datenseiten in den Pufferpool kann die Leistung verbessert werden, indem die E/A-Wartezeit verringert wird. Darüber hinaus wird die Effizienz des Vorablesezugriffs durch parallele E/A-Operationen erhöht.

Es gibt zwei Kategorien des Vorablesezugriffs:

- **Sequenzieller Vorablesezugriff:** Ein Mechanismus, mit dem aufeinander folgende Seiten in den Pufferpool gelesen werden, bevor die Seiten von der Anwendung angefordert werden.
- **Vorablesezugriff über Listen:** Wird manchmal als *sequenzieller Vorablesezugriff über Listen* bezeichnet. Ist eine effiziente Methode zum Vorablesen von Datenseiten, die nicht aufeinander folgen.

Diese beiden Methoden zum Lesen von Datenseiten erfolgen zusätzlich zum normalen Lesen. Normale Lesevorgänge werden verwendet, wenn nur eine oder wenige aufeinander folgende Seiten abgerufen werden. Bei einem normalen Lesevorgang wird jeweils nur eine Seite von Daten übertragen.

Vorablesezugriff und partitionsinterne Parallelität

Das Vorablesen ist wichtig für die Leistung der partitionsinternen Parallelität, bei der mehrere Subagenten beim Durchsuchen eines Index oder einer Tabelle verwendet werden. Solche parallelen Suchoperationen ziehen einen größeren Datenverarbeitungsdurchsatz nach sich, wodurch höhere Vorablesegeschwindigkeiten erforderlich werden.

Der Nachteil durch ungeeignetes Vorablesen ist bei parallelen Suchoperationen höher als bei seriellen Suchoperationen. Wenn für eine serielle Suchoperation kein Vorablesezugriff stattfindet, arbeit die Abfrage langsamer, weil der Agent immer auf E/A-Operationen warten muss. Wenn für eine parallele Suchoperation kein Vorablesezugriff stattfindet, müssen eventuell alle Subagenten warten, weil ein Subagent auf eine E/A-Operation wartet.

Aufgrund seiner Bedeutung wird der Vorablesezugriff bei partitionsinterner Parallelität intensiver durchgeführt. Die Funktion zur Sequenzerkennung toleriert größere Lücken zwischen benachbarten Seiten, so dass die Seiten als sequenziell betrachtet werden können. Die Breite dieser Lücken erhöht sich mit der Anzahl der an der Suchoperation beteiligten Subagenten.

Zugehörige Konzepte:

- „Pufferpoolverwaltung" auf Seite 257
- „Sequenzielles Vorablesen" auf Seite 269
- „Vorablesezugriff über Listen" auf Seite 272
- „Konfiguration von E/A-Servern für Vorablesezugriff und Parallelität" auf Seite 273
- „Illustration des Vorablesens mit paralleler E/A" auf Seite 275

Sequenzielles Vorablesen

Durch das Lesen mehrerer aufeinander folgender Seiten in den Pufferpool in einer einzigen E/A-Operation kann der Systemaufwand für Ihre Anwendung wesentlich reduziert werden. Darüber hinaus können mehrere parallele E/A-Operationen zum Lesen mehrerer Bereiche von Seiten in den Pufferpool bei der Verringerung der E/A-Wartezeiten helfen.

Der Vorablesezugriff beginnt, wenn der Datenbankmanager bestimmt, dass sequenzielle E/A-Operationen zweckmäßig sind und dass durch den Vorablesezugriff die Leistung verbessert werden könnte. In solchen Fällen wie Tabellensuchen und Sortierungen in Tabellen kann der Datenbankmanager leicht feststellen, dass durch den sequenziellen Vorablesezugriff die E/A-Leistung verbessert wird. In diesen Fällen startet der Datenbankmanager den sequenziellen Vorablesezugriff automatisch. Das folgende Beispiel zeigt eine Abfrage, die wahrscheinlich eine Tabellensuche erforderlich macht und für die deshalb der sequenzielle Vorablesezugriff die nahe liegende Methode wäre:

```
SELECT NAME FROM EMPLOYEE
```

Auswirkungen des PREFETCHSIZE-Werts für Tabellenbereiche

Zur Definition der Anzahl vorabgelesener Seiten für jeden Tabellenbereich können Sie die Klausel PREFETCHSIZE in den Anweisungen CREATE TABLESPACE oder ALTER TABLESPACE verwenden. Der von Ihnen definierte Wert wird in der Spalte PREFETCHSIZE der Systemkatalogtabelle SYSCAT.TABLESPACES gespeichert.

Es empfiehlt sich, den Wert für PREFETCHSIZE explizit als ein Vielfaches der Anzahl von Tabellenbereichsbehältern, der Anzahl physischer Platten unter jedem Behälter (wenn eine RAID-Einheit eingesetzt wird) und des Werts für EXTENTSIZE für Ihren Tabellenbereich zu definieren. Der Wert für EXTENTSIZE gibt die Anzahl von Seiten an, die der Datenbankmanager in einen Behälter schreibt, bevor er einen anderen Behälter verwendet. Ist zum Beispiel für EXTENTSIZE ein Wert von 16 Seiten festgelegt und hat der Tabellenbereich zwei Behälter, könnten Sie den Wert für die Anzahl der vorab gelesenen Seiten (PREFETCHSIZE) auf den Wert 32 setzen. Wenn fünf physische Platten pro Behälter vorhanden sind, können Sie die Menge der vorab gelesenen Daten auf 160 Seiten setzen.

Der Datenbankmanager überwacht die Verwendung des Pufferpools, um sicherzustellen, dass durch den Vorablesezugriff keine Seiten aus dem Pufferpool entfernt werden, wenn eine andere Arbeitseinheit sie noch benötigt. Zur Vermeidung von Problemen kann der Datenbankmanager die Anzahl der vorabgelesenen Seiten auf einen kleineren als den von Ihnen für den Tabellenbereich angegebenen Wert begrenzen.

Die Vorablesegröße (PREFETCHSIZE) kann erhebliche Auswirkungen auf die Leistung insbesondere für Suchoperationen in großen Tabellen haben. Verwenden Sie den Datenbanksystemmonitor und andere Systemmonitortools als Unterstützung bei der Optimierung des PREFETCHSIZE-Werts für Ihre Tabellenbereiche. Zum Beispiel könnten Sie folgende Arten von Informationen sammeln:

- Mit Überwachungsprogrammen für Ihr Betriebssystem können Sie feststellen, ob E/A-Wartezeiten für die Abfrage auftreten.
- Mit Hilfe des Datenelements *pool_async_data_reads (asynchrone Leseoperationen für Pufferpooldaten)*, das vom Datenbanksystemmonitor bereitgestellt wird, können Sie feststellen, ob Vorablesezugriffe stattfinden.

Wenn E/A-Wartezeiten auftreten und für die Abfrage der Vorablesezugriff aktiv ist, könnten Sie den Wert für PREFETCHSIZE erhöhen. Wenn die Vorablesefunktion nicht die Ursache für die E/A-Wartezeiten ist, verbessert eine Erhöhung des Werts für PREFETCHSIZE die Leistung Ihrer Abfrage nicht.

Bei allen Arten des Vorablesezugriffs können mehrere E/A-Operationen parallel ausgeführt werden, wenn für PREFETCHSIZE ein Vielfaches des Werts für EXTENTSIZE für den Tabellenbereich angegeben ist und sich die durch EXTENTSIZE definierten Bereiche des Tabellenbereichs in separaten Behältern befinden. Konfigurieren Sie die Behälter so, dass sie getrennte physische Einheiten verwenden, um eine bessere Leistung zu erzielen.

Sequenzerkennung

In einigen Fällen ist es nicht von vornherein offensichtlich, dass durch einen Vorablesezugriff eine Leistungsverbesserung erreicht werden kann. In diesen Fällen kann der Datenbankmanager die E/A-Operationen überwachen und den Vorablesezugriff aktivieren, wenn sequenzielles Lesen von Seiten auftritt. In solchen Fällen wird der Vorablesezugriff vom Datenbankmanager nach Zweckmäßigkeit aktiviert und inaktiviert. Diese Art des sequenziellen Vorablesens wird als *Sequenzerkennung* bezeichnet und wird für Index- und Datenseiten angewandt. Mit dem Konfigurationsparameter *seqdetect* können Sie steuern, ob der Datenbankmanager mit Sequenzerkennung arbeiten soll.

Wenn die Sequenzerkennung zum Beispiel aktiviert ist, könnte die folgende SQL-Anweisung von einem sequenziellen Vorablesezugriff profitieren:

```
SELECT NAME FROM EMPLOYEE
WHERE EMPNO BETWEEN 100 AND 3000
```

In diesem Beispiel könnte das Optimierungsprogramm vielleicht begonnen haben, die Tabelle mit Hilfe eines Index für die Spalte EMPNO zu durchsuchen. Wenn die Tabelle eine hohe Clusterbildung in Bezug auf diesen Index aufweist, dann sind die Leseoperationen für die Datenseiten beinahe sequenziell und ein Vorablesezugriff könnte zu einer Leistungsverbesserung führen. Infolgedessen findet ein Vorablesezugriff auf die Datenseiten statt.

In diesem Beispiel könnte auch ein Vorablesezugriff auf die Indexseiten erfolgen. Wenn eine große Anzahl von Indexseiten untersucht werden muss und der Datenbankmanager erkennt, dass ein sequenzielles Lesen der Indexseiten stattfindet, wird ein Vorablesezugriff auf Indexseiten durchgeführt.

Zugehörige Konzepte:

- „Pufferpoolverwaltung" auf Seite 257
- „Vorablesen von Daten in den Pufferpool" auf Seite 268
- „Vorablesezugriff über Listen" auf Seite 272
- „Blockorientierte Pufferpools für besseren sequenziellen Vorablesezugriff" auf Seite 271

Blockorientierte Pufferpools für besseren sequenziellen Vorablesezugriff

Das Vorablesen von Seiten vom Plattenspeicher ist aufgrund des E/A-Systemaufwands teuer. Der Durchsatz kann beträchtlich verbessert werden, wenn sich die Verarbeitung mit den E/A-Operationen überlappen kann. Die meisten Plattformen stellen Hochleistungsmechanismen bereit, die zusammenhängende Seiten vom Plattenspeicher in nicht zusammenhängende Bereiche des Hauptspeichers einlesen. Diese Mechanismen werden gewöhnlich als gestreutes Lesen (engl. *scattered read*) oder *über einen Vektor definierte E/A* bezeichnet. Auf einigen Plattformen kann die Leistung dieser Mechanismen nicht mit E/A-Operationen mit großen Blockgrößen konkurrieren.

Standardmäßig sind Pufferpools seitenorientiert. Dies bedeutet, dass zusammenhängende Seiten auf dem Plattenspeicher in nicht zusammenhängende Seiten im Arbeitsspeicher vorab gelesen werden. Der sequenzielle Vorablesezugriff kann verbessert werden, wenn zusammenhängende Seiten vom Plattenspeicher in zusammenhängende Seiten in einem Pufferpool gelesen werden können.

Zu diesem Zweck können Sie blockorientierte Pufferpools erstellen. Ein blockorientierter Pufferpool besteht sowohl aus einem Seitenbereich als auch aus einem Blockbereich. Der Seitenbereich ist für nicht sequenzielle Vorableseoperationen erforderlich. Der Blockbereich besteht aus Blöcken, wobei jeder Block eine angegebene Anzahl zusammenhängender Seiten enthält, die als *Blockgröße* bezeichnet wird.

Die optimale Nutzung eines blockorientierten Pufferpools hängt von der angegebenen Blockgröße ab. Die Blockgröße ist die Granularität, mit der E/A-Server bei sequenziellen Vorablesezugriffen eine blockorientierte E/A-Operation in Betracht ziehen. Die EXTENTSIZE-Größe ist die Granularität, mit der Tabellenbereiche in Behältern einheitenübergreifend gespeichert werden (Striping). Da mehrere Tabellenbereiche mit unterschiedlichen EXTENTSIZE-Werten an einen Pufferpool gebunden werden können, der mit der gleichen Blockgröße definiert ist, beachten Sie, in welcher Beziehung die EXTENTSIZE-Größe und die Blockgröße bei der effizienten Nutzung des Pufferpoolspeichers zueinander stehen. Pufferpoolspeicher kann unter folgenden Umständen verschwendet werden:

- Wenn der Wert für EXTENTSIZE, durch den die Größe für Vorableseanforderungen festgelegt wird, kleiner als der für den Pufferpool angegebene Wert für BLOCK_SIZE ist.
- Wenn sich einige Seiten, die durch eine Vorableseanforderung angefordert werden, bereits im Seitenbereich des Pufferpools befinden.

Der E/A-Server lässt einige verschwendete Seiten in jedem Pufferpoolblock zu. Wenn jedoch zu große Teile eines Blocks verschwendet würden, führt der E/A-Server ein nicht blockorientiertes Vorablesen in den Seitenbereich des Pufferpools durch. Dies führt nicht zu einer optimalen Leistung.

Zur Erzielung einer optimalen Leistung binden Sie Tabellenbereiche der gleichen EXTENTSIZE-Größe an einen Pufferpool mit einer Blockgröße, die der EXTENTSIZE-Größe der Tabellenbereiche entspricht. Eine gute Leistung lässt sich erreichen, wenn der Wert für EXTENTSIZE größer als die Blockgröße ist, jedoch nicht, wenn der Wert für EXTENTSIZE kleiner als die Blockgröße ist.

Verwenden Sie die Anweisungen CREATE und ALTER BUFFERPOOL zur Erstellung eines blockorientierten Pufferpools. Für blockorientierte Pufferpools gelten die folgenden Einschränkungen:

- Ein Pufferpool kann nicht blockorientiert sein und gleichzeitig den erweiterten Speicher verwenden.
- Blockorientierte E/A und AWE-Unterstützung können von einem Pufferpool nicht gleichzeitig genutzt werden. Die AWE-Unterstützung erhält Vorrang vor der Unterstützung für blockorientierte E/A, wenn beide für einen Pufferpool angegeben sind. In diesem Fall wird die Unterstützung für blockorientierte E/A für den Pufferpool inaktiviert. Sie wird wieder aktiviert, wenn die AWE-Unterstützung inaktiviert wird.

Anmerkung: Blockorientierte Pufferpools sind für den sequenziellen Vorablesezugriff gedacht. Wenn Ihre Anwendungen keinen sequenziellen Vorablesezugriff nutzen, ist der Blockbereich im Pufferpool verschwendet.

Zugehörige Konzepte:

- „Pufferpoolverwaltung" auf Seite 257
- „Vorablesen von Daten in den Pufferpool" auf Seite 268
- „Sequenzielles Vorablesen" auf Seite 269

Vorablesezugriff über Listen

Der *Vorablesezugriff über Listen* oder *sequenzielle Vorablesezugriff über Listen* ist eine effiziente Methode, auf Daten zuzugreifen, auch wenn die benötigten Datenseiten nicht in aufeinander folgender Reihenfolge vorliegen. Der Vorablesezugriff über Listen kann in Verbindung mit dem Zugriff über einen oder mehrere Indizes verwendet werden.

Wenn das Optimierungsprogramm einen Index zum Zugriff auf Zeilen verwendet, kann es das Lesen der Datenseiten verzögern, bis alle Zeilen-IDs (RIDs) aus dem Index empfangen wurden. Zum Beispiel könnte das Optimierungsprogramm eine Indexsuche durchführen, um die abzurufenden Zeilen und Datenseiten zu ermitteln, wenn zuvor der Index IX1 definiert wurde:

```
INDEX IX1:  NAME    ASC,
            DEPT    ASC,
            MGR     DESC,
            SALARY  DESC,
            YEARS   ASC
```

Betrachten Sie folgende Suchbedingungen:

```
WHERE NAME BETWEEN 'A' and 'I'
```

Wenn die Daten in Bezug auf diesen Index keine Clusterbildung aufweisen, enthält der Vorablesezugriff über Listen einen Schritt zum Sortieren der durch die Indexsuche ermittelten Liste von Zeilen-IDs (RIDs).

Zugehörige Konzepte:

- „Pufferpoolverwaltung" auf Seite 257
- „Vorablesen von Daten in den Pufferpool" auf Seite 268
- „Sequenzielles Vorablesen" auf Seite 269

E/A-Verwaltung

Dieser Abschnitt beschreibt die Optimierung von E/A-Servern.

Konfiguration von E/A-Servern für Vorablesezugriff und Parallelität

Zur Aktivierung des Vorablesezugriffs startet der Datenbankmanager zum Lesen von Datenseiten separate Steuerthreads, die als *E/A-Server* bezeichnet werden. Infolgedessen gliedert sich die Verarbeitung einer Abfrage in zwei parallele Aktivitäten: Datenverarbeitung (CPU) und E/A-Operationen für Datenseiten. Die E/A-Server warten auf Vorableseanforderungen aus der CPU-Verarbeitungsaktivität. Diese Vorableseanforderungen enthalten eine Beschreibung der E/A-Operationen, die zur Erfüllung der Abfrage benötigt werden. Die möglichen Vorablesemethoden bestimmen, wann und wie der Datenbankmanager die Vorableseanforderungen generiert.

Durch Konfigurieren einer ausreichenden Anzahl von E/A-Servern mit Hilfe des Konfigurationsparameters *num_ioservers* kann die Leistung von Abfragen, für die der Vorablesezugriff auf Daten verwendet werden kann, erheblich gesteigert werden. Setzen Sie den Wert des Parameters *num_ioservers* mindestens auf die Zahl der physischen Platten in der Datenbank, um die Möglichkeit für parallele E/A-Operationen zu maximieren.

Es ist besser, die Anzahl von E/A-Servern zu hoch als zu niedrig anzusetzen. Wenn Sie zusätzliche E/A-Server angeben, werden diese Server nicht verwendet, und ihre Speicherseiten werden ausgelagert. Infolgedessen wird die Leistung nicht beeinträchtigt. Jeder E/A-Serverprozess hat eine Nummer. Der Datenbankmanager verwendet immer den Prozess mit der niedrigsten Nummer, so dass einige der Prozesse mit höheren Nummern möglicherweise nie zum Einsatz kommen.

Bei der Schätzung, wie viele E/A-Server eventuell erforderlich sind, sollten Sie folgende Punkte beachten:

- Die Anzahl der Datenbankagenten, die Vorableseanforderungen an die E/A-Serverwarteschlange gleichzeitig schreiben könnten.
- Der höchste Grad, bis zu dem die E/A-Server parallel arbeiten können.

Konfiguration für asynchrone E/A

Auf einigen Plattformen arbeitet DB2® mit asynchronen E/A-Operationen (AIO), um die Leistung solcher Aktivitäten wie Seitenlöschfunktionen und Vorablesezugriffe zu verbessern. Asynchrone E/A (AIO) ist am effektivsten, wenn die Daten in den Behältern über mehrere Platten verteilt sind. Die Leistung profitiert außerdem von einer Optimierung der AIO-Infrastruktur des zugrunde liegenden Betriebssystems.

Unter AIX® können Sie zum Beispiel die AIO für das Betriebssystem optimieren. Wenn die asynchrone Ein-/Ausgabe entweder mit SMS-Behältern oder mit DMS-Dateibehältern arbeitet, verwalten Betriebssystemprozesse, so genannte AIO-Server, die E/A-Operationen. Eine kleine Anzahl solcher Server kann den Vorteil der asynchronen Ein-/Ausgabe einschränken, indem sie die Anzahl von AIO-Anforderungen begrenzt. Verwenden Sie zur Konfiguration der Anzahl von AIO-Servern unter AIX die *smit*-AIO-Parameter *minservers* und *maxservers*.

Zugehörige Konzepte:

- „Parallelverarbeitung für Anwendungen" auf Seite 103
- „Illustration des Vorablesens mit paralleler E/A" auf Seite 275
- „Verwaltung der parallelen Ein-/Ausgabe" auf Seite 276

Zugehörige Referenzen:

- „num_ioservers - Anzahl von E/A-Servern" auf Seite 440

Illustration des Vorablesens mit paralleler E/A

Die folgende Abbildung veranschaulicht, wie E/A-Server zum Vorablesen von Daten in den Pufferpool verwendet werden.

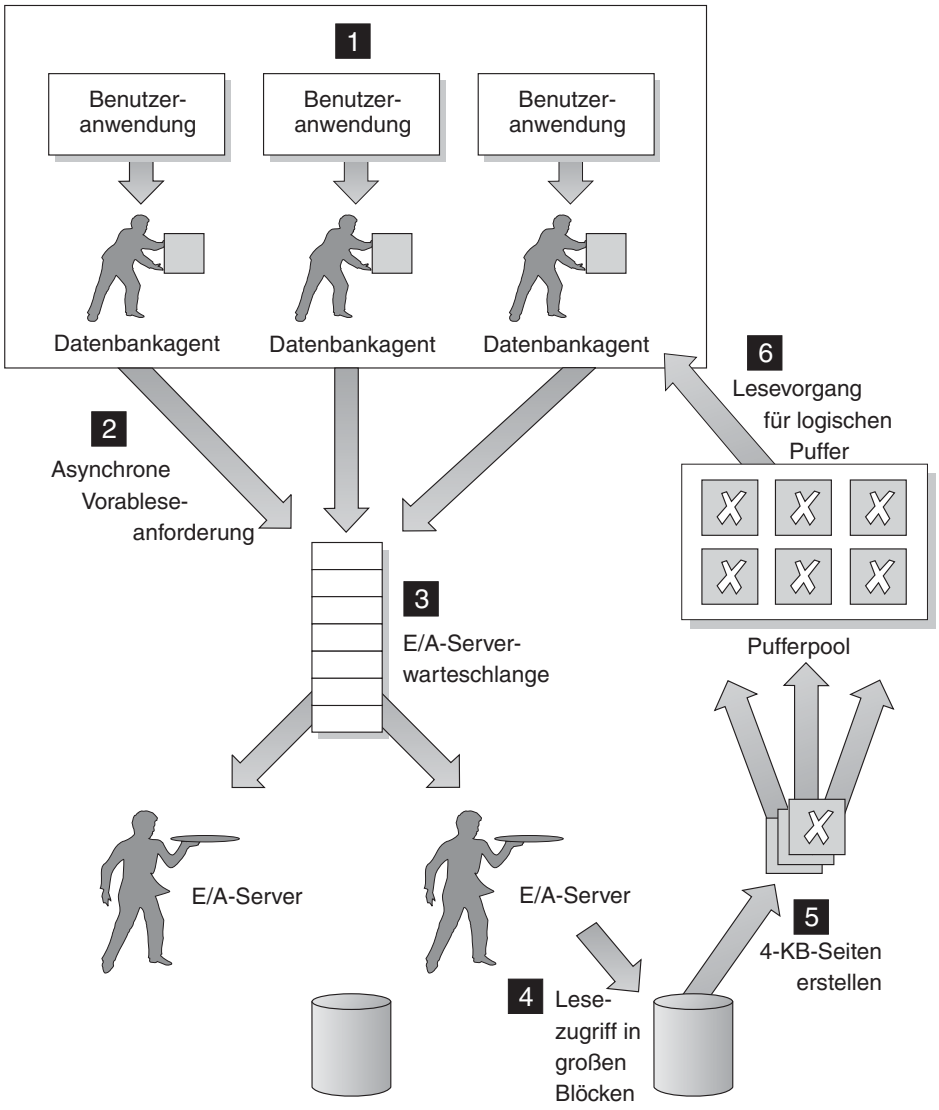


Abbildung 24. Vorablesen von Daten mit Hilfe von E/A-Servern

1 Die Benutzeranwendung übergibt die SQL-Anforderung an den Datenbankagenten, der der Benutzeranwendung vom Datenbankmanager zugeordnet wurde.

2 , **3**

Der Datenbankagent stellt fest, dass ein Vorablesezugriff erfolgen soll, um die angeforderten Daten abzurufen und so die SQL-Anforderung zu erfüllen, und schreibt eine Vorableseanforderung an die E/A-Server-warteschlange.

4, **5**

Der erste verfügbare E/A-Server liest die Vorableseanforderung aus der Warteschlange und liest die Daten aus dem Tabellenbereich in den Pufferpool. Die Anzahl von E/A-Servern, die Daten gleichzeitig aus einem Tabellenbereich abrufen können, hängt von der Anzahl der Vorleseanforderungen in der Warteschlange sowie von der Anzahl von E/A-Servern ab, die durch den Konfigurationsparameter *num_ioservers* der Datenbank konfiguriert sind.

6 Der Datenbankagent führt die erforderlichen Operationen an den Datenseiten im Pufferpool durch und liefert das Ergebnis an die Benutzeranwendung zurück.

Zugehörige Konzepte:

- „Vorablesen von Daten in den Pufferpool" auf Seite 268
- „Sequenzielles Vorablesen" auf Seite 269
- „Vorablesezugriff über Listen" auf Seite 272
- „Konfiguration von E/A-Servern für Vorablesezugriff und Parallelität" auf Seite 273
- „Verwaltung der parallelen Ein-/Ausgabe" auf Seite 276
- „Agenten in einer partitionierten Datenbank" auf Seite 308

Verwaltung der parallelen Ein-/Ausgabe

Wenn mehrere Behälter für einen Tabellenbereich vorhanden sind, kann der Datenbankmanager die *parallele Ein-/Ausgabe* einleiten, bei der der Datenbankmanager mehrere E/A-Server zur Verarbeitung der E/A-Anforderungen einer einzelnen Abfrage verwendet. Jeder E/A-Server verarbeitet die E/A-Operationen für einen anderen Behälter, so dass mehrere Behälter parallel gelesen werden können. Die parallele Durchführung der E/A-Operationen kann zu bedeutenden Verbesserungen beim E/A-Durchsatz führen.

Obwohl ein getrennter E/A-Server die E/A-Operationen für jeden Behälter durchführen kann, ist die tatsächliche Anzahl der E/A-Server, die parallele E/A-Operationen durchführen können, auf die Anzahl der physischen Einheiten begrenzt, über die die angeforderten Daten verteilt sind. Aus diesem Grund benötigen Sie so viele E/A-Server, wie physische Einheiten vorhanden sind.

Die parallele Ein-/Ausgabe wird in folgenden Fällen unterschiedlich eingeleitet:

- **Sequenzieller Vorablesezugriff**

  Beim sequenziellen Vorablesezugriff wird die parallele E/A initialisiert, wenn der Wert für PREFETCHSIZE (Menge der vorab gelesenen Daten) ein Vielfaches des Werts für EXTENTSIZE eines Tabellenbereichs ist. Jede Vorableseanforderung wird dann in viele kleine Anforderungen aufgeteilt, die sich an den Grenzen der durch EXTENTSIZE definierten Bereiche orientieren. Diese kleinen Anforderungen werden dann verschiedenen E/A-Servern zugeordnet.
- **Vorablesezugriff über Listen**

  Beim Vorablesezugriff über Listen wird jede Liste von Seiten in Abhängigkeit von den Behältern, in denen die Datenseiten gespeichert sind, in kleinere Listen unterteilt. Diese kleineren Listen werden dann verschiedenen E/A-Servern zugeordnet.

- **Sicherung und Wiederherstellung von Datenbanken oder Tabellenbereichen**

  Für Sicherungen oder Wiederherstellungen von Daten ist die Anzahl paralleler E/A-Anforderungen gleich der Größe des Sicherungspuffers dividiert durch den Wert von EXTENTSIZE. Der Maximalwert ist gleich der Anzahl von Behältern.
- **Wiederherstellung einer Datenbank oder eines Tabellenbereichs**

  Für die Wiederherstellung von Daten werden parallele E/A-Anforderungen in gleicher Weise wie beim sequenziellen Vorablesezugriff eingeleitet und aufgeteilt. Die Daten werden nicht im Pufferpool wiederhergestellt, sondern direkt aus dem Wiederherstellungspuffer auf die Platte versetzt.
- **LOAD**

  Beim Laden von Daten können Sie den Grad der E/A-Parallelität mit der Option DISK_PARALLELISM des Befehls LOAD angeben. Wenn diese Option nicht angegeben wird, verwendet der Datenbankmanager einen Standardwert, der auf der kumulativen Anzahl von Tabellenbereichsbehältern für alle Tabellenbereiche basiert, die der Tabelle zugeordnet sind.

Für eine optimale Leistung bei paralleler E/A sollten Sie folgende Voraussetzungen erfüllen:

- Es ist eine ausreichende Anzahl E/A-Server vorhanden. Geben Sie geringfügig mehr E/A-Server an als die Anzahl von Behältern, die für alle Tabellenbereiche in der Datenbank verwendet werden.
- Die Werte für EXTENTSIZE und PREFETCHSIZE sind für den Tabellenbereich angemessen. Zur Vermeidung einer übermäßigen Nutzung des Pufferpools sollte der Wert für PREFETCHSIZE nicht zu groß sein. Eine ideale Größe ist ein Vielfaches des Werts für EXTENTSIZE, der Anzahl physischer Platten unter jedem Behälter (wenn eine RAID-Einheit eingesetzt wird) und der Anzahl der Tabellenbereichsbehälter. Der Wert für EXTENTSIZE sollte relativ klein sein, wobei sich ein Wert zwischen 8 und 32 Seiten empfiehlt.
- Die Behälter befinden sich auf separaten physischen Laufwerken.
- Alle Behälter haben dieselbe Größe, um einen konsistenten Grad an Parallelität zu gewährleisten.

  Wenn ein oder mehrere Behälter kleiner als die anderen sind, verringern sie das Potenzial für das optimierte parallele Vorablesen. Betrachten Sie die folgenden Beispiele:

  – Wenn ein kleinerer Behälter gefüllt ist, werden weitere Daten in den übrigen Behältern gespeichert, wodurch sich eine ungleichmäßige Auslastung der Behälter ergibt. Ungleichmäßig ausgelastete Behälter beeinträchtigen die Leistung des parallelen Vorablesens, da die Anzahl von Behältern, aus denen Daten vorab gelesen werden können, eventuell kleiner ist als die Gesamtanzahl von Behältern.
  – Wenn ein kleinerer Behälter zu einem späteren Zeitpunkt hinzugefügt wird und die Daten neu verteilt werden, enthält der kleinere Behälter weniger Daten als die anderen Behälter. Diese im Verhältnis zu den anderen Behältern kleine Menge von Daten führt nicht zu einer Optimierung des parallelen Vorablesens.
  – Wenn ein Behälter größer ist und alle anderen Behälter vollständig gefüllt werden, wird dieser Behälter zum einzigen Behälter, in dem weitere Daten gespeichert werden. Beim Zugriff auf diese weiteren Daten kann der Datenbankmanager keinen parallelen Vorablesezugriff durchführen.

- Es ist eine angemessene E/A-Kapazität vorhanden, wenn partitionsinterne Parallelität verwendet wird. Auf SMP-Maschinen kann die partitionsinterne Parallelität die für eine Abfrage benötigte Zeit reduzieren, indem die Abfrage auf mehreren Prozessoren ausgeführt wird. Es ist eine ausreichende E/A-Kapazität erforderlich, um jeden Prozessor auszulasten. In der Regel sind zusätzliche physische Laufwerke erforderlich, um diese E/A-Kapazität bereitzustellen.

  Der Wert für PREFETCHSIZE muss höher sein, um ein Vorablesen mit höheren Geschwindigkeiten und eine effektive Nutzung der E/A-Kapazität zu ermöglichen.

  Die Anzahl der erforderlichen physischen Laufwerke hängt von der Geschwindigkeit und der Kapazität der Laufwerke und des E/A-Busses sowie von der Geschwindigkeit der Prozessoren ab.

Zugehörige Konzepte:

- „Konfiguration von E/A-Servern für Vorablesezugriff und Parallelität" auf Seite 273
- „Illustration des Vorablesens mit paralleler E/A" auf Seite 275
- „Richtlinien für die Sortierleistung" auf Seite 278

Richtlinien für die Sortierleistung

Da Abfragen häufig sortierte oder gruppierte Ergebnisse erfordern, werden häufig Sortierungen angefordert, und eine geeignete Konfiguration der Sortierspeicherbereiche spielt für eine gute Abfrageleistung eine wichtige Rolle. Sortieren ist in folgenden Fällen erforderlich:

- Es gibt keinen Index, um eine angeforderte Reihenfolge (z. B. durch eine Anweisung SELECT mit einer Klausel ORDER BY) der Daten herzustellen.
- Es gibt einen Index, aber Sortieren ist effizienter als der Zugriff über den Index.
- Ein Index wird erstellt.
- Ein Index wird gelöscht, wodurch eine Sortierung der Indexseitennummern verursacht wird.

Ein Sortiervorgang umfasst zwei Phasen:

1. Die Sortierphase

   Eine Sortierung kann *mit* oder *ohne* Überlauf stattfinden. Wenn die sortierten Daten nicht vollständig in den Sortierzwischenspeicher, das heißt in den Speicherblock, der jedes Mal zugeordnet wird, wenn eine Sortierung durchgeführt wird, passen, laufen die Daten in temporäre Datenbanktabellen über. Sortierungen, die keinen Überlauf verursachen, zeigen stets eine bessere Leistung als solche, bei denen ein Überlauf auftritt.
2. Die Rückgabe der Ergebnisse der Sortierphase

   Die Rückgabe kann mit *Piping* oder *ohne* Piping erfolgen. Wenn sortierte Daten direkt zurückgegeben werden können, ohne dass eine temporäre Tabelle zum Speichern der endgültigen sortierten Liste von Daten erforderlich ist, handelt es sich um eine Sortiervorgang mit Piping (d. h. die Daten werden über eine Pipe zurückgegeben). Wenn die sortierten Daten über eine temporäre Tabelle zurückgegeben werden müssen, wird dies als Sortierung ohne Piping bezeichnet. Eine Sortierung mit Piping ist immer effizienter als eine Sortierung ohne Piping.

Elemente mit Auswirkung auf das Sortieren

Die folgenden Elemente wirken sich auf die Sortierleistung aus:

- Die Einstellungen für die folgenden Datenbankkonfigurationsparameter:
  - Die Zwischenspeichergröße für Sortierlisten (*sortheap*) gibt die Größe des Speichers an, der für jede Sortierung verwendet wird.
  - Der Schwellenwert für Sortierspeicher (*sheapthres*) und der Schwellenwert für Sortierspeicher für gemeinsame Sortiervorgänge (*sheapthres_shr*) steuern die Gesamtgröße des Speichers, der für das Sortieren im gesamten Exemplar für alle Sortiervorgänge verfügbar ist.
- Anweisungen, die eine große Menge Sortierungen verursachen
- Fehlende Indizes, die zur Vermeidung unnötiger Sortiervorgänge dienen könnten
- Anwendungslogik, die das Sortieraufkommen nicht minimiert
- Paralleles Sortieren, das die Leistung der Sortierungen erhöht, aber nur auftreten kann, wenn die Anweisung partitionsinterne Parallelität verwendet

Im Allgemeinen sollte der für das gesamte Exemplar verfügbare Sortierspeicher (*sheapthres*) so groß wie möglich sein, ohne übermäßiges Seitenauslagern (Paging) zu verursachen. Obwohl eine Sortierung vollständig im Sortierspeicher durchgeführt werden kann, kann dies zu einem übermäßigen Auslagern von Seiten führen. In diesem Fall geht der Vorteil eines großen Sortierzwischenspeichers verloren. Aus diesem Grund sollten Sie einen Betriebssystemmonitor verwenden, um alle Änderungen in der Auslagerung von Seiten durch das System zu verfolgen, wenn Sie die Konfigurationsparameter für das Sortieren anpassen.

Beachten Sie außerdem, dass bei einer Sortierung mit Piping der Sortierzwischenspeicher nicht freigegeben wird, bevor die Anwendung den dieser Sortierung zugeordneten Cursor schließt. Eine Sortierung mit Piping kann weiter Speicher belegen, bis der Cursor geschlossen wird.

Anmerkung: Nachdem die DB2®-Binärsortierung mit Teilschlüssel verbessert wurde, so dass auch nicht ganzzahlige Datentypschlüssel unterstützt werden, ist beim Sortieren langer Schlüssel zusätzlicher Speicherplatz erforderlich. Wenn lange Schlüssel für Sortierungen verwendet werden, erhöhen Sie den Wert für den Konfigurationsparameter *sortheap*.

Techniken zur Verwaltung der Sortierleistung

Ermitteln Sie bestimmte Anwendungen und Anweisungen, bei denen die Sortierung ein wesentliches Leistungsproblem darstellt:

- Richten Sie Ereignismonitore (Event Monitors) auf Anwendungs- und Anweisungsebene ein, um Unterstützung bei der Ermittlung von Anwendungen mit der längsten Gesamtsortierzeit zu erhalten.
- Ermitteln Sie innerhalb dieser Anwendungen die Anweisungen mit der längsten *Gesamtsortierzeit*.
- Optimieren Sie diese Anweisungen mit Hilfe eines Programms wie Visual Explain.
- Stellen Sie sicher, dass geeignete Indizes vorhanden sind. Sie können mit Hilfe von Visual Explain alle Sortieroperationen für eine bestimmte Anweisung ermitteln. Stellen Sie anschließend fest, ob ein geeigneter Index für jede Tabelle vorhanden ist, auf die durch diese Anweisung zugegriffen wird.

Anmerkung: Sie können die EXPLAIN-Tabellen durchsuchen, um die Abfragen mit Sortieroperationen zu ermitteln.

Sie können den Datenbanksystemmonitor und Vergleichstesttechniken bei der Einstellung der Konfigurationsparameter *sortheap* und *sheapthres* zur Unterstützung heranziehen. Gehen Sie für jeden Datenbankmanager und die zugehörigen Datenbanken folgendermaßen vor:

- Erstellen Sie eine repräsentative Auslastung und führen Sie sie aus.
- Sammeln Sie für jede betroffene Datenbank Durchschnittswerte für die folgenden Leistungsvariablen über den Auslastungszeitraum der Vergleichstests:
  - Gesamter verwendeter Sortierspeicher
  - Aktive Sortiervorgänge
- Setzen Sie den Parameter *sortheap* auf den Durchschnittswert für den *gesamten verwendeten Sortierspeicher* für jede Datenbank.
- Definieren Sie den Wert für *sheapthres*. Gehen Sie wie folgt vor, um eine angemessene Größe zu schätzen:
  1. Stellen Sie fest, welche Datenbank in dem Exemplar über den größten Wert für *sortheap* verfügt.
  2. Ermitteln Sie die durchschnittliche Größe des Sortierspeichers für diese Datenbank.

     Wenn die Ermittlung des Durchschnittswerts zu aufwendig ist, verwenden Sie als Wert 80% des maximalen Sortierspeichers.
  3. Setzen Sie den Wert für *sheapthres* auf die Durchschnittsanzahl der aktiven Sortiervorgänge multipliziert mit der oben berechneten Durchschnittsgröße des Sortierspeichers.

     Dies ist die empfohlene Anfangseinstellung. Anschließend können Sie mit Hilfe von Vergleichstests diesen Wert optimieren.

Zugehörige Referenzen:

- „sortheap - Zwischenspeichergröße für Sortierlisten" auf Seite 418
- „sheapthres - Schwellenwert für Sortierspeicher" auf Seite 416
- „sheapthres_shr - Schwellenwert für Sortierspeicher für gemeinsame Sortiervorgänge" auf Seite 405

Tabellenverwaltung

Dieser Abschnitt beschreibt die Methoden zur Verwaltung von Tabellen im Hinblick auf Leistungsverbesserungen.

Tabellenreorganisation

Nach zahlreichen Änderungen an Tabellendaten können sich logisch sequenzielle Daten auf physisch nicht sequenziellen Datenseiten befinden, so dass der Datenbankmanager zusätzliche Leseoperationen ausführen muss, um auf die Daten zuzugreifen. Darüber hinaus sind zusätzliche Leseoperationen auch erforderlich, wenn eine beträchtliche Anzahl von Zeilen gelöscht wurde. In einem solchen Fall können Sie eine Reorganisation der Tabelle in Betracht ziehen, um die Tabelle mit dem Index in Übereinstimmung zu bringen und Speicherplatz wieder verfügbar zu machen. Sie können sowohl die Systemkatalogtabellen als auch Datenbanktabellen reorganisieren.

Anmerkung: Da die Reorganisation einer Tabelle in der Regel mehr Zeit als das Erfassen von Statistikdaten beansprucht, können Sie das Dienstprogramm RUNSTATS ausführen, um die Statistiken für Ihre Daten auf den aktuellen Stand zu bringen, und Ihre Anwendungen erneut binden. Wenn aktualisierte Statistiken keine Leistungsverbesserungen erzielen, kann eine Reorganisation eventuell helfen. Detaillierte Informationen zu den Optionen und der Funktionsweise des Dienstprogramms REORG TABLE finden Sie in der zugehörigen Befehlsreferenz.

Beachten Sie die folgenden Faktoren, die Hinweise darauf geben können, dass eine Tabelle reorganisiert werden sollte:

- Es ist ein hohes Aufkommen an INSERT-, UPDATE- und DELETE-Aktivitäten an Tabellen zu verzeichnen, auf die durch Abfragen zugegriffen wird.
- Es treten wesentliche Änderungen in der Leistung von Abfragen auf, die einen Index mit einem hohen Clusterverhältnis verwenden.
- Die Leistung wird durch die Ausführung von RUNSTATS zur Aktualisierung der Statistikdaten nicht besser.
- Der Befehl REORGCHK zeigt den Bedarf einer Reorganisation der Tabelle an.
- Die Kosten einer zunehmenden Verschlechterung der Abfrageleistung und die Kosten einer Reorganisation der Tabelle, zu denen die CPU-Zeit, die zur Ausführung erforderliche Zeit sowie der verringerte Grad des gemeinsamen Zugriffs aufgrund der Sperre, die das Dienstprogramm REORG für die Tabelle bis zum Ende der Reorganisation aktiviert, zu rechnen sind, müssen gegeneinander abgewogen werden.

Verringern der Erforderlichkeit der Reorganisation von Tabellen

Damit die Reorganisation einer Tabelle weniger häufig erforderlich wird, führen Sie nach dem Erstellen der Tabelle folgende Aufgaben aus:

- Ändern Sie die Tabelle (ALTER), um PCTFREE hinzuzufügen.
- Erstellen Sie einen Clusterindex mit PCTFREE für Index.
- Sortieren Sie die Daten.
- Laden Sie die Daten.

Wenn Sie diese Maßnahmen durchgeführt haben, tragen der Clusterindex der Tabelle und die Einstellung von PCTFREE für die Tabelle dazu bei, die ursprüngliche Sortierreihenfolge der Daten beizubehalten. Wenn genügend Speicherplatz auf den Tabellenseiten freigehalten wird, können neue Daten auf den richtigen Seiten eingefügt werden, um die Clusterbildung in Bezug auf den Index beizubehalten. Mit wachsender Menge eingefügter Daten und zunehmender Füllung der Seiten der Tabelle werden Datensätze an das Ende der Tabelle angehängt, so dass die Tabelle allmählich ihre Clusterbildung verliert.

Wenn Sie das Dienstprogramm REORG TABLE ausführen oder eine Sortierung und ein Laden von Daten nach der Erstellung eines Clusterindex durchführen, versucht der Index eine bestimmte Ordnung der Daten zu erhalten, wodurch sich die Werte für die Statistiken CLUSTERRATIO bzw. CLUSTERFACTOR verbessern, die durch das Dienstprogramm RUNSTATS erfasst werden.

Anmerkung: Die Erstellung von Tabellen mit mehrdimensionalem Clustering (MDC) kann die Erforderlichkeit der Reorganisation von Tabellen senken. Bei MDC-Tabellen bleibt die Clusterbildung für die Spalten erhalten, die Sie als Argumente der Klausel ORGANIZE BY DIMENSIONS in der Anweisung CREATE TABLE angeben. Allerdings kann das Dienstprogramm REORGCHK eine Reorganisation einer MDC-Tabelle empfehlen, wenn es ermittelt, dass zu viele ungenutzte Blöcke vorhanden sind oder dass Blöcke zusammengelegt werden sollten.

Zugehörige Konzepte:

- „Indexreorganisation" auf Seite 298
- „Informationen zu DMS-Einheiten" auf Seite 301
- „SMS-Tabellenbereiche" auf Seite 16
- „DMS-Tabellenbereiche" auf Seite 17
- „Tabellen- und Indexverwaltung für Standardtabellen" auf Seite 20
- „Snapshot monitor" in *System Monitor Guide and Reference*
- „Tabellen- und Indexverwaltung für MDC-Tabellen" auf Seite 24

Zugehörige Tasks:

- „Feststellen des Zeitpunkts zur Reorganisation von Tabellen" auf Seite 282
- „Auswählen einer Methode zur Tabellenreorganisation" auf Seite 286

Feststellen des Zeitpunkts zur Reorganisation von Tabellen

Normale Datenbankaktivitäten wie zum Beispiel wiederholte Einfüge-, Lösch- und Aktualisierungsoperationen können sich mit der Zeit auf die Organisation der Daten in Ihren Tabellen und Indizes auswirken und die Datenbankleistung beeinträchtigen. Das Dienstprogramm erfasst Statistiken zur Organisation der Daten in Ihren Datenbanktabellen, so dass Sie statistische Informationen über die Tabellengröße und die Datenverteilung, das Clusterverhältnis (Cluster ratio) von Indizes, die Anzahl von Blattseiten (Leaf pages) in Indizes, die Anzahl von Tabellenzeilen, die aus ihren ursprünglichen Seiten überlaufen, sowie die Anzahl gefüllter und leerer Seiten in einer Tabelle ermitteln können. Darüber hinaus können Sie mit Hilfe von RUNSTATS auch Informationen zur Effizienz des Vorablesezugriffs (Prefetching) sammeln.

Die von RUNSTATS erfassten Statistikdaten werden in den Systemkatalogtabellen gespeichert. Diese Informationen können Sie bei der Entscheidung unterstützen, wann Tabellen und Indizes zu reorganisieren sind und welcher Typ von Reorganisation erforderlich ist.

Neben der Verwendung von RUNSTATS zur Erfassung von Informationen zu Ihrer Datenbankorganisation zu einem bestimmten Zeitpunkt empfiehlt es sich auch, RUNSTATS regelmäßig auszuführen, um allgemeine Trends zu ermitteln, die mit Änderungen in der Datenbankleistung verbunden sein können.

Anmerkung: Der Befehl REORGCHK liefert ebenfalls Informationen zur Datenorganisation und kann Sie beraten, ob bestimmte Tabellen reorganisiert werden müssen. Allerdings kann die Ausführung spezieller Abfragen auf die Katalogstatistiktabellen in regelmäßigen Intervallen ein Leistungsdatenprotokoll liefern, das es Ihnen ermöglicht, Trends auszumachen, die eine größere Tragweite für die Leistung haben.

Vorgehensweise:

Fragen Sie die Katalogstatistiken ab und überwachen Sie die folgenden Statistiken, um zu bestimmen, ob Sie Tabellen reorganisieren müssen:

1. Überlauf von Zeilen

   Fragen Sie die Spalte OVERFLOW in der Tabelle SYSSTAT.TABLES ab, um den OVERFLOW-Wert zu überwachen. Die Werte in dieser Spalte stellen die Anzahl von Zeilen dar, die nicht auf ihre ursprünglichen Seiten passen. Zeilendaten können Überlaufen, wenn VARCHAR-Spalten mit Werten aktualisiert werden, die länger als die ersten Werte sind. In diesen Fällen wird an der ursprünglichen Stelle der Zeile ein Zeiger hinterlegt, und der tatsächliche Wert an einer anderen, von dem Zeiger angegebenen Stelle gespeichert. Dadurch kann die Leistung beeinträchtigt werden, da der Datenbankmanager dem Zeiger folgen muss, um den Inhalt der Zeile zu finden. Durch diese beiden Schritte verlängert sich die Verarbeitungszeit und erhöht sich möglicherweise auch die Anzahl der erforderlichen E/A-Operationen.

   Eine Reorganisation der Tabellendaten beseitigt die Zeilenüberläufe. Daher erhöht sich mit steigender Anzahl von Überlaufzeilen auch der potenzielle Nutzen einer Reorganisation der Tabellendaten.

2. Abrufstatistiken (Fetch)

   Fragen Sie die drei folgenden Spalten in den Katalogstatistiktabellen SYSCAT.INDEXES und SYSSTAT.INDEXES ab, um die Effektivität der Vorablesefunktionen beim Zugriff auf die Tabellen in der Indexreihenfolge zu ermitteln. Diese Statistiken vermitteln einen Eindruck von der durchschnittlichen Leistung der Vorablesefunktionen für die zugrunde liegende Tabelle.

   - In der Spalte AVERAGE_SEQUENCE_FETCH_PAGES wird die durchschnittliche Anzahl von Seiten gespeichert, auf die in der Reihenfolge in der Tabelle zugegriffen werden kann. Die Seiten, auf die in der Reihenfolge zugegriffen werden kann, kommen für den Vorablesezugriff in Betracht. Ein kleiner Wert gibt an, dass die Vorablesefunktionen nicht so effizient sind, wie sie sein könnten, weil sie nicht die gesamte Anzahl von Seiten, die durch die Einstellung PREFETCHSIZE für den Tabellenbereich definiert sind, einlesen können. Ein hoher Wert zeigt an, dass die Vorablesefunktionen effektiv arbeiten. Für eine Tabelle mit Clusterindex sollte dieser Wert dem Wert für NPAGES nahe kommen, der die Anzahl von Seiten angibt, die Zeilen enthalten.
   - In der Spalte AVERAGE_RANDOM_FETCH_PAGES wird die durchschnittliche Anzahl von Tabellenseiten gespeichert, auf die beim Abrufen von Tabellenzeilen über den Index zwischen sequenziellen Seitenzugriffen ein wahlfreier Zugriff erfolgt. Die Vorablesefunktionen ignorieren kleine Anzahlen wahlfreier Seiten, wenn die Mehrzahl der Seiten in der Reihenfolge vorliegt, und setzen den Vorablesezugriff bis zur konfigurierten Vorablesegröße fort. Mit zunehmender Unordnung in der Reihenfolge der Tabelle steigt die Anzahl von Seiten, auf die ein wahlfreier Zugriff erfolgen muss. Eine solche zunehmende Unordnung wird in der Regel durch Einfügungen außerhalb der Reihenfolge, das heißt, entweder am Ende der Tabelle oder in Überlauf-

seiten, verursacht. Dies führt zu Abrufen, die die Abfrageleistung verlangsamen, wenn der Index für den Zugriff auf einen Bereich von Werten verwendet wird.

- In der Spalte AVERAGE_SEQUENCE_FETCH_GAP wird die durchschnittliche Lücke zwischen Tabellenseiten in der Reihenfolge beim Abrufen über den Index gespeichert. Erkannt beim Durchsuchen von Indexseiten stellt jede Lücke die durchschnittliche Anzahl von Seiten dar, die jeweils zwischen Seiten in der Reihenfolge wahlfrei abgerufen werden müssen. Solche Lücken treten auf, wenn auf viele Seiten wahlfrei zugegriffen wird, wodurch die Vorablesefunktionen angehalten werden. Ein hoher Wert zeigt an, dass eine Tabelle nicht gut organisiert oder bezüglich des Index eine niedrige Clusterbildung aufweist.

3. Anzahl von Indexblattseiten, die als gelöscht markierte, jedoch noch nicht gelöschte Satz-IDs (RIDs) enthalten

   In Indizes des Typs 2 werden Satz-IDs in der Regel nicht physisch gelöscht, wenn die Satz-ID als gelöscht markiert wird. Dies bedeutet, dass nützlicher Speicherplatz von diesen logisch gelöschten Satz-IDs belegt sein könnte. Zum Abrufen der Anzahl von Blattseiten, auf denen jede Satz-ID als gelöscht markiert ist, fragen Sie die Spalte NUM_EMPTY_LEAFS der Statistiktabellen SYSCAT.INDEXES und SYSSTAT.INDEXES ab. Für Blattseiten, in denen nicht alle Satz-IDs als gelöscht markiert sind, wird die Gesamtzahl logisch gelöschter Satz-IDs in der Spalte NUMRIDS_DELETED gespeichert.

   Verwenden Sie diese Informationen zur Abschätzung, wie viel Speicherplatz durch eine Ausführung des Befehls REORG INDEXES mit der Option CLEANUP ALL zurückgewonnen werden könnte. Um nur den Speicherplatz in Seiten wieder verfügbar zu machen, in denen alle Satz-IDs als gelöscht markiert sind, führen Sie den Befehl REORG INDEXES mit der Option CLEANUP ONLY PAGES aus.

4. Statistiken zum Clusterverhältnis und zum Clusterfaktor für Indizes

   In der Spalte CLUSTERRATIO der Katalogtabelle SYSTCAT.INDEXES wird ein Statistikwert für das Clusterverhältnis gespeichert. Dieser Wert (zwischen 0 und 100) stellt den Grad der Datenclusterbildung bezüglich des Index dar. Wenn Sie detaillierte Indexstatistikdaten (DETAILED) erfassen, wird ein feinerer Statistikwert zwischen 0 und 1 für den Clusterfaktor in der Spalte CLUSTERFACTOR gespeichert und der Wert der Spalte CLUSTERRATIO ist –1. Nur einer dieser beiden Werte zur Clusterbildung wird im Katalog SYSCAT.INDEXES aufgezeichnet. Zum Vergleich der CLUSTERFACTOR-Werte mit den CLUSTERRATIO-Werten müssen Sie den CLUSTERFACTOR-Wert mit 100 multiplizieren, um einen Prozentwert zu erhalten.

   Anmerkung: Im Allgemeinen kann nur einer der Indizes in einer Tabelle einen hohen Grad an Clusterbildung aufweisen.

   Indexsuchen, die nicht mit einem reinen Indexzugriff durchgeführt werden, erzielen bei höheren Clusterverhältnissen wahrscheinlich eine bessere Leistung. Ein niedriges Clusterverhältnis führt zu vermehrten Ein-/Ausgabeoperationen für diese Art der Suche, da nach dem ersten Zugriff auf eine Datenseite die Wahrscheinlichkeit geringer ist, dass sich diese Seite immer noch im Pufferpool befindet, wenn der nächste Zugriff auf sie erfolgt. Die Leistung für einen Index ohne Clusterbildung kann möglicherweise durch Erhöhen der Puffergröße verbessert werden.

Wenn Tabellendaten anfangs bezüglich eines bestimmten Index Clusterbildung aufwiesen und die Statistikdaten zur Clusterbildung nun anzeigen, dass in Bezug auf denselben Index nur eine geringe Clusterbildung vorhanden ist, kann es sinnvoll sein, die Tabelle zu reorganisieren, um die Daten bezüglich dieses Index wieder in Clustern anzuordnen.

5. Anzahl der Blattseiten (Leaf pages)

   Fragen Sie die Spalte NLEAF in der Tabelle SYSCAT.INDEXES ab, um die Anzahl von Blattseiten zu ermitteln, die von einem Index belegt werden. Diese Anzahl gibt Auskunft darüber, wie viele E/A-Operationen für Indexseiten für ein komplettes Durchsuchen eines Index erforderlich sind.

   Ein Index sollte idealerweise möglichst wenig Speicherplatz belegen, um die E/A-Operationen für eine Indexsuche zu reduzieren. Wahlfreie Aktualisierungen können dazu führen, dass Seiten geteilt werden und sich der Index dadurch vergrößert. Wenn Indizes während der Reorganisation einer Tabelle neu erstellt werden, kann jeder Index mit dem minimal erforderlichen Speicherbereich erstellt werden.

   Anmerkung: Bei der Erstellung von Indizes werden standardmäßig zehn Prozent des freien Speicherbereichs auf jeder Indexseite nicht belegt. Zur Erhöhung des Betrages an freiem Speicherbereich geben Sie den Parameter PCTFREE bei der Erstellung des Index an. Der Wert für PCTFREE wird bei jeder Reorganisation des Index verwendet. Ein freier Speicherbereich von mehr als zehn Prozent kann die Häufigkeit der Indexreorganisation verringern, weil in dem zusätzlichen Speicherbereich weitere Indexeinfügungen untergebracht werden können.

6. Vergleich von Dateiseiten

   Zur Berechnung der Anzahl leerer Seiten in einer Tabelle fragen Sie die Spalten FPAGES und NPAGES der Tabelle SYSCAT.TABLES ab und subtrahieren den NPAGES-Wert vom FPAGES-Wert. In der Spalte FPAGES wird die Gesamtanzahl der verwendeten Seiten, in der Spalte NPAGES die Anzahl von Seiten gespeichert, die Zeilen enthalten. Leere Seiten können auftreten, wenn ganze Bereiche von Zeilen gelöscht werden.

   Mit dem Ansteigen der Anzahl leerer Seiten wächst die Notwendigkeit einer Reorganisation für eine Tabelle. Bei der Reorganisation einer Tabelle werden die leeren Seiten aus dem Speicherbereich für eine Tabelle entfernt und der von der Tabelle belegte Speicherplatz verringert. Da leere Seiten bei einer Tabellensuche auch in den Pufferpool gelesen werden, kann durch das Entfernen ungenutzter Seiten die Leistung einer Tabellensuche verbessert werden.

Zugehörige Konzepte:

- „Katalogstatistiktabellen" auf Seite 126
- „Tabellenreorganisation" auf Seite 280
- „Indexreorganisation" auf Seite 298

Zugehörige Tasks:

- „Erfassen von Katalogstatistiken" auf Seite 117

Auswählen einer Methode zur Tabellenreorganisation

DB2® stellt zwei Methoden zur Reorganisation von Tabellen bereit: die klassische und die ″am Platz″ (engl. ″in place″). Im Allgemeinen ist die klassische Tabellenreorganisation schneller, sollte jedoch nur verwendet werden, wenn Ihre Anwendungen während der Reorganisation ohne Schreibzugriff auf Tabellen funktionieren. Wenn Ihre Umgebung diese Einschränkung nicht zulässt, kann eine Reorganisation am Platz durchgeführt werden, die zwar langsamer ist, jedoch im Hintergrund stattfinden kann, während der normale Datenzugriff fortgesetzt wird. Machen Sie sich mit den Merkmalen jeder Methode vertraut und entscheiden Sie, welche Methode für Ihre Umgebung geeigneter ist.

Vorgehensweise:

Beachten Sie bei der Auswahl einer Methode zur Tabellenreorganisation die Merkmale der folgenden Methoden:

- **Klassische Tabellenreorganisation**

  Diese Methode bietet die schnellste Tabellenreorganisation, insbesondere wenn Sie keine LOB- oder LONG-Daten reorganisieren müssen. Darüber hinaus werden Indizes in perfekter Reihenfolge nach der Reorganisation der Tabelle neu erstellt. Reine Leseanwendungen können auf die Originalkopie der Tabelle außer während der letzten Phasen der Reorganisation zugreifen. Dies sind die Phasen, in denen die Spiegelkopie der Tabelle durch die permanente Tabelle ersetzt und die Indizes neu erstellt werden.

  Auf der anderen Seite sind die folgenden möglichen Nachteile zu berücksichtigen:

  – Große Speicherplatzanforderungen

    Da die klassische Tabellenreorganisation eine Spiegelkopie der Tabelle erstellt, kann sie das Zweifache des Speicherplatzes der ursprünglichen Tabelle beanspruchen. Wenn die reorganisierte Tabelle größer als die ursprüngliche Tabelle ist, kann die Reorganisation mehr als das Doppelte des Platzes der ursprünglichen Tabelle benötigen.

    Die Spiegelkopie kann in einem temporären Tabellenbereich erstellt werden, wenn der Tabellenbereich nicht groß genug ist, jedoch erreicht die Ersetzungsphase ihre beste Leistung, wenn sie im gleichen DMS-Tabellenbereich stattfindet. Tabellen in SMS-Tabellenbereichen müssen die Spiegelkopie immer in einem temporären Tabellenbereich speichern.

  – Begrenzter Tabellenzugriff

    Auch ein reiner Lesezugriff ist auf die ersten Phasen des Reorganisationsprozesses beschränkt.

  – Alles-oder-Nichts-Prozess

    Wenn die Reorganisation an einem Punkt fehlschlägt, muss sie auf den Knoten, auf denen sie fehlgeschlagen ist, völlig neu gestartet werden.

  – Ausführung innerhalb des Controllers der Anwendung, die sie aufruft

    Die Reorganisation kann nur durch diese Anwendung bzw. durch einen Benutzer gestoppt werden, der weiß, wie der Prozess zu stoppen ist und die Berechtigung zur Ausführung des Befehls FORCE (erzwungenes Abbrechen) für die Anwendung hat.

  Empfehlung: Wählen Sie diese Methode, wenn Sie für die Reorganisation von Tabellen ein Wartungszeitfenster zur Verfügung haben.

- **Tabellenreorganisation am Platz**

  Die Am-Platz-Methode arbeitet langsamer und gewährleistet keine perfekt angeordneten Daten, jedoch kann sie Anwendungen den Zugriff auf die Tabelle während der Reorganisation ermöglichen. Darüber hinaus kann eine Tabellenreorganisation am Platz von einem Benutzer angehalten und fortgesetzt werden, der die entsprechende Berechtigung hat, indem er den Schemanamen und den Tabellennamen verwendet.

  Anmerkung: Die Tabellenreorganisation am Platz ist nur für Tabellen mit Indizes des Typs 2 und ohne erweiterte Indizes zulässig.

  Ziehen Sie die folgenden Nachteile in Betracht:

  – Keine perfekte Indexreorganisation

    Sie müssen Indizes eventuell später reorganisieren, um die Indexfragmentierung zu verringern und Indexobjektspeicherplatz wieder verfügbar zu machen.

  – Längere Ausführungszeit

    Falls erforderlich, gibt die Reorganisation am Platz gleichzeitigen Anwendungen Vorrang. Dies bedeutet, dass lang laufende Anweisungen oder RR- und RS-Leseoperationen in lang laufenden Anwendungen den Reorganisationsprozess bremsen können. Eine Reorganisation am Platz läuft in einer OLTP-Umgebung eventuell schneller, in der viele kleine Transaktionen ausgeführt werden.

  – Höhere Protokollspeicheranforderungen

    Da eine Tabellenreorganisation am Platz ihre Aktivitäten protokolliert, um eine Wiederherstellung nach einem unerwarteten Fehler zu ermöglichen, erfordert sie mehr Protokollspeicherplatz als eine klassische Reorganisation.

    Es ist möglich, dass eine Reorganisation am Platz einen Protokollspeicher in mehrfacher Größe der reorganisierten Tabelle benötigt. Die Größe des erforderlichen Speicherplatzes hängt von der Anzahl der Zeilen, die versetzt werden, sowie von der Anzahl und Größe der Indizes für die Tabelle ab.

  Empfehlung: Wählen Sie die Tabellenreorganisation am Platz für Umgebungen im 24x7-Betrieb mit minimalen Wartungsfenstern aus.

Detaillierte Informationen zur Ausführung dieser Methoden zur Tabellenreorganisation finden Sie in den Beschreibungen der Syntax des Befehls REORG TABLE.

Überwachen des Fortschritts der Tabellenreorganisation

Informationen über den aktuellen Fortschritt der Tabellenreorganisation werden in die Protokolldatei für Datenbankaktivitäten geschrieben. Die Protokolldatei enthält einen Datensatz für jedes Reorganisationsereignis. Zum Anzeigen dieser Datei führen Sie den Befehl `db2 list history` für die Datenbank aus, in der sich die Tabelle befindet, die reorganisiert wird.

Sie können den Fortschritt der Tabellenreorganisation auch mit Hilfe von Tabellenmomentaufnahmen überwachen. Überwachungsdaten zur Tabellenreorganisation werden unabhängig von der Einstellung des Datenbankmonitorschalters für Tabellen aufgezeichnet.

Wenn ein Fehler auftritt, wird ein SQLCA-Auszug in die Protokolldatei geschrieben. Bei einer Tabellenreorganisation am Platz wird der Status mit PAUSED (angehalten) aufgezeichnet.

Zugehörige Konzepte:

- „Tabellenreorganisation" auf Seite 280
- „Indexreorganisation" auf Seite 298

Zugehörige Tasks:

- „Feststellen des Zeitpunkts zur Reorganisation von Tabellen" auf Seite 282

Indexverwaltung

Die folgenden Abschnitte beschreiben die Indexreorganisation zur Erzielung von Leistungsverbesserungen.

Vor- und Nachteile von Indizes

Obwohl das Optimierungsprogramm entscheidet, ob ein Index für den Zugriff auf Tabellendaten verwendet werden soll, müssen Sie, abgesehen von den folgenden Ausnahmen, ermitteln, welche Indizes die Leistung verbessern könnten, und diese Indizes erstellen. Ausnahmen sind die Dimensionsblockindizes und die zusammengesetzten Blockindizes, die für jede Dimension automatisch erstellt werden, die Sie bei der Erstellung einer MDC-Tabelle (MDC - Multi-Dimensional Clustering) angeben.

Sie müssen außerdem in folgenden Situationen das Dienstprogramm RUNSTATS ausführen, um neue Statistikdaten zu den Indizes zu erfassen:

- Nachdem Sie einen Index erstellt haben.
- Nachdem Sie den Wert für PREFETCHSIZE geändert haben.

Darüber hinaus sollten Sie das Dienstprogramm RUNSTATS in regelmäßigen Abständen ausführen, um die Statistikdaten auf aktuellem Stand zu halten. Ohne aktuelle Statistiken zu Indizes ist das Optimierungsprogramm nicht in der Lage, den besten Datenzugriffsplan für Abfragen zu ermitteln.

Anmerkung: Um festzustellen, ob ein Index in einem bestimmten Paket verwendet wird, verwenden Sie die SQL-EXPLAIN-Einrichtung. Zum Planen von Indizes können Sie sich mit dem Designadvisor in der Steuerzentrale oder mit dem Tool `db2advis` Hinweise zu Indizes geben lassen, die von einer oder mehreren SQL-Anweisungen verwendet werden könnten.

Vorteile eines Index im Gegensatz zu keinem Index

Wenn für eine Tabelle kein Index vorhanden ist, muss eine Tabellensuche für jede Tabelle durchgeführt werden, auf die in einer Datenbankabfrage verwiesen wird. Je umfangreicher die Tabelle ist, desto länger dauert die Tabellensuche, weil eine Tabellensuche erfordert, dass auf jede Tabellenzeile sequenziell zugegriffen wird. Obwohl eine Tabellensuche für eine komplexe Abfrage, welche die meisten der Zeilen in einer Tabelle abruft, effizienter sein kann, ist für eine Abfrage, die nur einige Tabellenzeilen liefert, eine Indexsuche für den Zugriff auf Tabellenzeilen effizienter.

Das Optimierungsprogramm wählt einen Index, wenn in der SELECT-Anweisung auf die Indexspalten verwiesen wird und wenn aufgrund der Schätzungen zu erwarten ist, dass eine Indexsuche schneller als eine Tabellensuche durchgeführt werden kann. Indexdateien sind im Allgemeinen kleiner und erfordern weniger Zeit zum Lesen als eine ganze Tabelle, besonders wenn die Tabellen umfangreicher werden. Darüber hinaus braucht eventuell nicht der gesamte Index durchsucht zu werden. Die Vergleichselemente, die auf einen Index angewandt werden, verringern die Anzahl der Zeilen, die aus den Datenseiten gelesen werden müssen.

Wenn eine angeforderte Sortierreihenfolge der Ausgabe einer Indexspalte entspricht, können die Zeilen durch eine Suche in diesem Index in der Spaltenreihenfolge gleich in der richtigen Reihenfolge ohne Sortierung abgerufen werden.

Jeder Indexeintrag enthält einen Suchschlüsselwert und einen Zeiger auf die Zeile, die den entsprechenden Wert enthält. Wenn Sie den Parameter ALLOW REVERSE SCANS in der Anweisung CREATE INDEX angeben, können die Werte sowohl in aufsteigender als auch in absteigender Reihenfolge gesucht werden. Es ist daher möglich, die Suche unter Angabe geeigneter Vergleichselemente zu begrenzen. Ein Index kann auch dazu verwendet werden, Zeilen in einer geordneten Reihenfolge abzurufen und somit zu vermeiden, dass der Datenbankmanager die Zeilen nach dem Lesen aus der Tabelle in einem weiteren Arbeitsgang sortieren muss.

Neben dem Suchschlüsselwert und dem Zeilenzeiger kann ein Index auch INCLUDE-Spalten enthalten, bei denen es sich um nicht indizierte Spalten in der indizierten Zeile handelt. Solche Spalten machen es dem Optimierungsprogramm möglich, die angeforderten Informationen aus dem Index abzurufen, ohne auf die Tabelle selbst zugreifen zu müssen.

Anmerkung: Das Vorhandensein eines Index für die Tabelle, die abgefragt wird, garantiert jedoch keine sortierte Ergebnismenge. Nur durch die Angabe der Klausel ORDER BY lässt sich die Reihenfolge der Ergebnismenge sicherstellen.

Obwohl Indizes die Zugriffszeiten erheblich verringern können, können sie auch nachteilige Auswirkungen auf die Leistung haben. Vor der Erstellung von Indizes sind daher die Auswirkungen mehrerer Indizes auf den Plattenspeicherplatz und die Verarbeitungszeit zu bedenken:

- Jeder Index erfordert Speicher oder Plattenspeicherplatz. Die exakte Menge ist von der Größe der Tabelle und der Größe und Anzahl von Spalten in dem Index abhängig.
- Jede für eine Tabelle ausgeführte Operation INSERT oder DELETE erfordert eine zusätzliche Aktualisierung aller Indizes für diese Tabelle. Dies gilt auch für jede Operation UPDATE, mit der der Wert eines Indexschlüssels geändert wird.
- Das Dienstprogramm LOAD erstellt alle vorhandenen Indizes neu bzw. hängt Daten an sie an.

  Der Parameter `indexfreespace` MODIFIED BY kann für den Befehl LOAD angegeben werden, um den Wert für PCTREE des Index außer Kraft zu setzen, der bei der Erstellung des Index verwendet wurde.
- Jeder Index macht potenziell einen alternativen Zugriffspfad für eine Abfrage verfügbar, den das Optimierungsprogramm zu berücksichtigen hat. Dadurch verlängert sich die Kompilierungszeit.

Wählen Sie Indizes sorgfältig, um den Anforderungen des Anwendungsprogramms gerecht zu werden.

Zugehörige Konzepte:

- „Speicherbedarf für Indizes" in *Systemverwaltung: Konzept*
- „Indexplanung - Tipps" auf Seite 290
- „Indexleistung - Tipps" auf Seite 293
- „Der Designadvisor" auf Seite 237
- „Tabellenreorganisation" auf Seite 280
- „Indexreorganisation" auf Seite 298
- „Tabellen- und Indexverwaltung für Standardtabellen" auf Seite 20
- „Tabellen- und Indexverwaltung für MDC-Tabellen" auf Seite 24
- „Indexbereinigung und Indexpflege" auf Seite 296

Zugehörige Tasks:

- „Erstellen eines Index" in *Systemverwaltung: Implementierung*
- „Erfassen von Katalogstatistiken" auf Seite 117
- „Erfassen von Indexstatistiken" auf Seite 119

Indexplanung - Tipps

Die Indizes, die Sie erstellen, sollten von den Daten und den Abfragen abhängig sein, die auf sie zugreifen.

Verwenden Sie den Designadvisor in der Steuerzentrale oder das Tool `db2advis`, um die besten Indizes für eine bestimmte Abfrage oder für eine Reihe von Abfragen, die eine Auslastung definieren, zu finden. Dieses Tool empfiehlt Indizes mit leistungssteigernden Merkmalen wie zum Beispiel INCLUDE-Spalten, übernommenen eindeutigen Indizes und mit ALLOW REVERSE SCANS definierten Indizes.

Die folgenden Richtlinien geben Anhaltspunkte zur Erstellung nützlicher Indizes für verschiedene Zwecke:

- Definieren Sie zur Vermeidung einiger Sortieroperationen an allen möglichen Stellen Primärschlüssel und eindeutige Schlüssel, indem Sie die Anweisung CREATE UNIQUE INDEX verwenden.
- Fügen Sie eindeutigen Indizes zur Verbesserung des Datenabrufs INCLUDE-Spalten hinzu. Zu diesem Zweck bieten sich Spalten mit folgenden Eigenschaften an:
  - Auf sie wird häufig zugegriffen, so dass die Leistung durch einen reinen Indexzugriff verbessert werden kann.
  - Sie sind zur Begrenzung des Bereichs einer Indexsuche nicht erforderlich.
  - Sie haben keinen Einfluss auf die Reihenfolge oder die Eindeutigkeit des Indexschlüssels.
- Verwenden Sie für einen effizienten Zugriff auf kleine Tabellen Indizes, um häufige Abfragen auf Tabellen mit einer größeren Anzahl von Datenseiten, wie in der Spalte NPAGES in der Katalogsicht SYSCAT.TABLES eingetragen, zu optimieren. Gehen Sie wie folgt vor:
  - Erstellen Sie einen Index für jede Spalte, die bei der Verknüpfung von Tabellen verwendet werden soll.
  - Erstellen Sie einen Index für jede Spalte, die regelmäßig nach bestimmten Werten durchsucht werden soll.
- Treffen Sie eine Entscheidung zwischen aufsteigender und absteigender Reihenfolge von Schlüsseln, je nachdem, welche Reihenfolge am häufigsten benötigt wird, um die Sucheffizienz zu erhöhen. Obwohl die Werte in umgekehrter Richtung gesucht werden können, wenn Sie den Parameter ALLOW REVERSE

SCANS in der Anweisung CREATE INDEX angeben, zeigen Suchoperationen in der angegebenen Indexreihenfolge eine etwas bessere Leistung als umgekehrte Suchen.

- Beachten Sie folgende Punkte, um den Aufwand der Indexverwaltung und den Indexspeicherbedarf möglichst gering zu halten:
  - Vermeiden Sie die Erstellung von Indizes, die aus einem Teil der Schlüssel anderer Indexschlüssel für die Spalten bestehen. Wenn es zum Beispiel einen Index für die Spalten a, b und c gibt, ist es im Allgemeinen nicht sinnvoll, einen zweiten Index für die Spalten a und b zu erstellen.
  - Erstellen Sie nicht willkürlich Indizes für alle Spalten. Unnötige Indizes belegen nicht nur Speicherplatz, sondern führen auch zu langen Vorbereitungszeiten (PREPARE). Dies spielt besonders für komplexe Abfragen eine wichtige Rolle, wenn eine Optimierungsklasse mit dynamisch programmierter Verknüpfungsaufzählung (Dynamic Programming Join Enumeration) verwendet wird.

    Beachten Sie die folgende allgemeine Regel für die typische Anzahl von Indizes, die Sie für eine Tabelle erstellen. Diese Anzahl richtet sich nach der primären Verwendung der Datenbank:
    - Erstellen Sie für Umgebungen zur Onlinetransaktionsverarbeitung (OLTP-Umgebungen) nur einen oder zwei Indizes.
    - Für Umgebungen mit reinen Leseabfragen können Sie mehr als fünf Indizes erstellen.
    - Für gemischte Abfrage- und OLTP-Umgebungen können Sie zwischen zwei und fünf Indizes erstellen.
- Erstellen Sie Indizes für Fremdschlüssel, um die Leistung von DELETE- und UPDATE-Operationen an der übergeordneten Tabelle zu verbessern.
- Erstellen Sie zur Verbesserung von DELETE- und UPDATE-Operationen mit gespeicherten Abfragetabellen (MQTs), die mit IMMEDIATE und INCREMENTAL definiert sind, eindeutige Indizes für den implizierten eindeutigen Schlüssel der gespeicherten Abfragetabelle, das heißt, für die Spalten in der GROUP BY-Klausel der MQT-Definition.
- Erstellen Sie Indizes für Spalten, die häufig zum Sortieren der Daten verwendet werden, um schnelle Sortieroperationen zu ermöglichen.
- Wenn Sie bei der Erstellung eines mehrspaltigen Index zur Verbesserung der Verknüpfungsleistung mehrere Auswahlmöglichkeiten für die erste Schlüsselspalte haben, verwenden Sie die Spalte, die am häufigsten in Verknüpfungen mit Gleichheitsvergleichselementen („=“) auftritt, oder die Spalte mit der größten Anzahl unterschiedlicher Werte als ersten Schlüssel.
- Definieren Sie einen Clusterindex, um neu eingefügte Zeilen nach diesem Index in Cluster einzureihen und Seitentrennungen zu vermeiden. Ein Clusterindex sollte die Notwendigkeit, die Tabelle zu reorganisieren, erheblich herabsetzen.

  Verwenden Sie bei der Definition einer Tabelle das Schlüsselwort PCTFREE, um anzugeben, wie viel freier Speicherplatz auf der Seite beibehalten werden soll, damit eingefügte Zeilen auf Seiten geeignet gespeichert werden können. Sie können auch die Klausel `pagefreespace` MODIFIED BY des Befehls LOAD angeben.
- Zur Aktivierung der Online-Indexdefragmentierung verwenden Sie der Erstellung von Indizes die Option MINPCTUSED. Mit der Option MINPCTUSED wird die Schwelle für die Mindestgröße des genutzten Speicherbereichs auf einer äußeren Indexseite (Blattseite) angegeben und die Online-Indexdefragmentierung aktiviert. Dadurch kann sich die Notwendigkeit für eine Reorganisation verringern. Dies geht jedoch auf Kosten einer Leistungseinbuße beim Löschen von Schlüsseln, wenn durch dieses Löschen Schlüssel physisch von der Indexseite entfernt werden.

Die Erstellung eines Index kommt unter folgenden Umständen in Betracht:

- Erstellen Sie einen Index für Spalten, die in Klauseln WHERE der am häufigsten verarbeiteten Abfragen und Transaktionen verwendet werden.

  Betrachten Sie zum Beispiel die folgende Klausel WHERE:

  ```
  WHERE WORKDEPT='A01' OR WORKDEPT='E21'
  ```

  Diese Klausel wird im Allgemeinen von einem Index für die Spalte WORKDEPT profitieren, sofern die Spalte WORKDEPT nicht viele doppelte Werte enthält.

- Erstellen Sie einen Index für eine Spalte bzw. Spalten, um die Zeilen in der von der Abfrage angeforderten Reihenfolge zu ordnen. Das Ordnen ist nicht nur für die Klausel ORDER BY erforderlich, sondern auch für andere Klauseln wie DISTINCT und GROUP BY.

  Im folgenden Beispiel wird die Klausel DISTINCT verwendet:

  ```
  SELECT DISTINCT WORKDEPT
    FROM EMPLOYEE
  ```

  Der Datenbankmanager kann einen Index verwenden, der in aufsteigender oder absteigender Reihenfolge für die Spalte WORKDEPT definiert ist, um doppelte Werte zu eliminieren. Derselbe Index könnte auch verwendet werden, um Werte wie in folgendem Beispiel mit einer Klausel GROUP BY zu gruppieren:

  ```
  SELECT WORKDEPT, AVERAGE(SALARY)
    FROM EMPLOYEE
  GROUP BY WORKDEPT
  ```

- Erstellen Sie einen Index mit einem zusammengesetzten Schlüssel, der jede Spalte nennt, auf die in einer Anweisung verwiesen wird. Wenn ein Index in dieser Weise angegeben wird, können Daten allein aus dem Index abgerufen werden, was effizienter ist, als auf eine Tabelle zuzugreifen.

  Betrachten Sie zum Beispiel die folgende SQL-Anweisung:

  ```
  SELECT LASTNAME
    FROM EMPLOYEE
    WHERE WORKDEPT IN ('A00','D11','D21')
  ```

  Wenn ein Index für die Spalten WORKDEPT und LASTNAME der Tabelle EMPLOYEE definiert ist, kann die Anweisung eventuell effizienter verarbeitet werden, indem nur der Index und nicht die gesamte Tabelle durchsucht wird. Beachten Sie, dass hier die Spalte WORKDEPT die erste Spalte des Index sein sollte, da sich das Vergleichselement auf diese Spalte bezieht.

- Erstellen Sie einen Index mit INCLUDE-Spalten, um die Verwendung von Indizes für Tabellen zu verbessern. Für das vorige Beispiel ließe sich ein eindeutiger Index folgendermaßen definieren:

  ```
  CREATE UNIQUE INDEX x ON employee (workdept) INCLUDE (lastname)
  ```

  Die Angabe von lastname als INCLUDE-Spalte und nicht als Teil des Indexschlüssels bedeutet, dass lastname nur auf den äußeren Seiten (Blattseiten) des Index gespeichert wird.

Zugehörige Konzepte:

- „Vor- und Nachteile von Indizes" auf Seite 288
- „Indexleistung - Tipps" auf Seite 293
- „Der Designadvisor" auf Seite 237
- „Indexreorganisation" auf Seite 298
- „Online-Indexdefragmentierung" auf Seite 300
- „Erstellung, Platzierung und Verwendung von Tabellen mit mehrdimensionalem Clustering (MDC)" in *Systemverwaltung: Konzept*

Indexleistung - Tipps

Beachten Sie die folgenden Vorschläge zur Verwendung und Verwaltung von Indizes:

- **Angeben der Parallelität bei Indexerstellung und -reorganisation**

  Bei der Erstellung bzw. Reorganisation von Indizes für umfangreiche Tabellen, die auf einer SMP-Maschine bereitgestellt werden, sollten Sie in Betracht ziehen, den Parameter *intra_parallel* auf 1 (YES) bzw. -1 (SYSTEM) zu setzen, um die Leistungsvorteile der Parallelverarbeitung zu nutzen.

  Zum Suchen und Sortieren von Daten können mehrere Prozessoren verwendet werden.

- **Angeben eines großen Dienstprogrammzwischenspeichers**

  Sowohl für die Indexerstellung (CREATE INDEX) als auch die Reorganisation von Indizes (REORG INDEXES) wird Schreibzugriff anderer Benutzer bzw. Anwendungen auf die zugrunde liegende Tabelle unterstützt. Wenn Sie ein hohes Aktualisierungsaufkommen an der zugrunde liegenden Tabelle für den Index erwarten, der erstellt oder reorganisiert wird, sollten Sie erwägen, einen großen Dienstprogrammzwischenspeicher zu konfigurieren. Ein großer Dienstprogrammzwischenspeicher beschleunigt die Erstellung bzw. Reorganisation der Indizes während der Übernahmephase. Alle Schreibvorgänge für die erstellten bzw. reorganisierten Indizes werden in den DB2®-Protokollen und im internen Speicherpufferbereich aufgezeichnet. Der interne Speicherpufferbereich ist ein ausgewiesener Speicherbereich, der bei Bedarf vom Dienstprogrammzwischenspeicher zugeordnet wird, um die Änderungen an den erstellten bzw. reorganisierten Indizes zu speichern. Die Verwendung dieses Speichers beschleunigt die Übernahmephase. Der zugeordnete Speicher wird nach Beendigung der Indexerstellung bzw. -reorganisation wieder freigegeben. Die Leistung während der Übernahmephase lässt sich stark verbessern, wenn sichergestellt wird, dass ein ausreichender Dienstprogrammzwischenspeicher zur Verfügung steht, um alle oder zumindest die meisten Änderungen an den erstellten bzw. reorganisierten Indizes aufzunehmen.

- **Angeben separater Tabellenbereiche für Indizes**

  Indizes können in einem anderen Tabellenbereich als die Tabellendaten gespeichert werden. Dadurch kann Plattenspeicher eventuell effizienter genutzt werden, indem die Bewegungen der Schreib-/Leseköpfe beim Indexzugriff reduziert werden. Sie können auch Indextabellenbereiche auf schnelleren physischen Einheiten erstellen. Darüber hinaus können Sie den Tabellenbereich für Indizes einem anderen Pufferpool zuordnen, wodurch die Indexseiten vielleicht länger im Puffer bleiben, weil sie nicht mit den Tabellendatenseiten konkurrieren.

  Wenn Sie Indizes nicht in getrennten Tabellenbereichen speichern, verwenden sowohl die Datenseiten als auch die Indexseiten dieselben Werte für EXTENTSIZE und PREFETCHSIZE. Wenn Sie für Indizes einen anderen Tabellenbereich verwenden, können Sie unterschiedliche Werte für alle Merkmale eines Tabellen-

bereichs auswählen. Da Indizes in der Regel kleiner als Tabellen sind und sich über weniger Behälter erstrecken, haben Indizes in der Regel auch kleinere Werte für EXTENTSIZE (z. B. 8 und 16 Seiten). Das SQL-Optimierungsprogramm zieht die Geschwindigkeit des Geräts für einen Tabellenbereich bei der Auswahl eines Zugriffsplans in Betracht.

- **Sicherstellen des Grads der Clusterbildung**

  Wenn Ihre SQL-Anweisung eine Reihenfolge zum Beispiel durch Klauseln wie ORDER BY, GROUP BY und DISTINCT anfordert und ein Index diese Anforderung erfüllt, kann das Optimierungsprogramm in folgenden Fällen entscheiden, den Index dennoch nicht zu verwenden:
  - Wenn der Grad der Index-Clusterbildung gering ist. Untersuchen Sie die Spalten CLUSTERRATIO und CLUSTERFACTOR des Katalogs SYSCAT.INDEXES, um Informationen zu erhalten.
  - Wenn die Tabelle so klein ist, dass es weniger aufwendig ist, die Tabelle zu durchsuchen und die Ergebnismenge im Hauptspeicher zu sortieren.
  - Wenn für den Zugriff auf die Tabelle konkurrierende Indizes gibt

  Führen Sie nach der Erstellung eines Clusterindex einen Befehl REORG TABLE im klassischen Modus aus, wodurch ein perfekt organisierter Index erstellt wird. Zur erneuten Herstellung einer Clusterbildung der Tabelle müssen Sie eventuell eine Sortieroperation oder eine LOAD-Operation durchführen. Im Allgemeinen ist die Clusterbildung in einer Tabelle nur anhand eines Index möglich. Erstellen Sie weitere Indizes, nachdem Sie den Clusterindex erstellt haben. Ein Clusterindex versucht, eine bestimmte Reihenfolge der Daten zu erhalten, wodurch die vom Dienstprogramm RUNSTATS gesammelten statistischen Werte für CLUSTERRATIO bzw. CLUSTERFACTOR verbessert werden.

  Zur Erhaltung des Clusterverhältnisses können Sie einen geeigneten Wert für PCTFREE beim Ändern einer Tabelle angeben, bevor Sie diese Tabelle mit Daten füllen (LOAD) oder reorganisieren. Der durch PCTFREE angegebene freie Speicherplatz auf jeder Seite reserviert Platz für Einfügungen, so dass diese Einfügungen der Clusterbildung entsprechend vorgenommen werden können. Wenn Sie PCTFREE für die Tabelle nicht angeben, wird der gesamte freie Speicherplatz durch die Reorganisation beseitigt.

  Anmerkung: Die Clusterbildung wird zurzeit bei Aktualisierungen (UPDATE) nicht beibehalten, sofern es sich nicht um Bereichsclustertabellen handelt. Das heißt, wenn Sie einen Datensatz aktualisieren, so dass sich der Schlüsselwert im Clusterindex ändert, wird dieser Datensatz nicht unbedingt auf eine entsprechend andere Seite versetzt, um die Clusterreihenfolge beizubehalten. Verwenden Sie zur Beibehaltung der Clusterbildung anstelle der Anweisung UPDATE die Anweisung DELETE und anschließend INSERT.

- **Aktualisieren der Tabellen- und Indexstatistiken**

  Führen Sie nach der Erstellung eines neuen Index das Dienstprogramm RUNSTATS aus, um Indexstatistikdaten zu sammeln. Anhand dieser Statistikdaten kann das Optimierungsprogramm bestimmen, ob durch die Verwendung des Index die Zugriffsleistung verbessert werden kann.

- **Aktivieren der Online-Indexdefragmentierung**

  Die Online-Indexdefragmentierung wird aktiviert, wenn die Klausel MINPCTUSED auf einen Wert größer null für den Index gesetzt wird. Die Online-Indexdefragmentierung ermöglicht eine Komprimierung von Indizes durch Zusammenfügen von Blattseiten, wenn der freie Speicherplatz auf einer Seite den angegebenen Wert erreicht oder unterschreitet, wobei der Index jedoch verfügbar bleibt.

- **Reorganisieren von Indizes nach Bedarf**

  Um optimale Leistungsvorteile aus den Indizes zu ziehen, sollten Sie eine regelmäßige Reorganisation der Indizes in Betracht ziehen, da Aktualisierungen an Tabellen dazu führen, dass der Vorablesezugriff auf Indexseiten an Effizienz einbüßt.

  Zur Reorganisation eines Index können Sie ihn entweder löschen und neu erstellen oder das Dienstprogramm REORG verwenden.

  Geben Sie zur Verringerung der Notwendigkeit für häufige Reorganisationen bei der Erstellung eines Index einen geeigneten Wert für PCTFREE an, um einen Prozentsatz an freiem Speicherplatz auf jeder Indexblattseite, wenn sie erstellt wird, freizuhalten. So ist bei zukünftigen Aktivitäten, durch die Datensätze in den Index eingefügt werden, die Wahrscheinlichkeit geringer, dass Indexseitentrennungen verursacht werden. Seitentrennungen bewirken, dass Indexseiten nicht mehr direkt aufeinander folgen oder sequenziell sind, was wiederum zu einer verminderten Effizienz des Vorablesezugriffs auf Indexseiten führt.

  Anmerkung: Der durch PCTFREE bei der Erstellung des Index angegebene Speicherplatz wird beibehalten, wenn der Index reorganisiert wird.

  Durch Löschen und erneutes Erstellen bzw. Reorganisieren des Index wird eine neue Menge von Seiten erstellt, die in etwa zusammenhängend und sequenziell sind und den Vorablesezugriff auf Indexseiten verbessern. Obwohl das Dienstprogramm REORG TABLE mehr Aufwand an Zeit und Ressourcen erfordert, stellt es sicher, dass die Datenseiten in Clustern angeordnet werden. Die Clusterbildung bietet größere Vorteile bei Indexsuchen, durch die auf eine bedeutende Anzahl von Datenseiten zugegriffen wird.

  In einer symmetrischen Multiprozessorumgebung (SMP) kann der „klassische" Modus des Dienstprogramms REORG TABLE, bei dem eine Spiegeltabelle zur schnellen Tabellenreorganisation verwendet wird, mehrere Prozessoren zur erneuten Erstellung von Indizes nutzen, sofern der Konfigurationsparameter *intra_parallel* des Datenbankmanagers den Wert YES oder ANY hat.

- **Analysieren von EXPLAIN-Informationen zur Indexnutzung**

  Führen Sie in regelmäßigen Abständen EXPLAIN für Ihre am häufigsten verwendeten Abfragen aus, und überprüfen Sie, ob jeder Ihrer Indizes wenigstens einmal verwendet wird. Wenn ein Index in keiner Abfrage verwendet wird, empfiehlt es sich, diesen Index zu löschen.

  EXPLAIN-Informationen geben Ihnen auch Einblick, ob Tabellensuchen in großen Tabellen als innere Tabellen bei Verknüpfungen mit Verschachtelungsschleife verarbeitet werden. Dies würde darauf hinweisen, dass ein Index für die Spalte des Verknüpfungsvergleichselements entweder fehlt oder als ineffektiv für die Anwendung des Verknüpfungsvergleichselements betrachtet wird.

- **Verwenden flüchtiger Tabellen für Tabellen mit stark variierender Größe**

  Eine *flüchtige* Tabelle ist eine Tabelle, deren zur Ausführungszeit zwischen leer und sehr groß schwanken kann. Für diese Art von Tabelle, bei der die Kardinalität stark variiert, kann das Optimierungsprogramm einen Zugriffsplan generieren, der eine Tabellensuche einer Index vorzieht.

  Durch die Deklaration einer Tabelle als „flüchtig" in der Anweisung ALTER TABLE...VOLATILE ist es dem Optimierungsprogramm möglich, eine Indexsuche auf der flüchtigen Tabelle auszuführen. In den folgenden Fällen verwendet das Optimierungsprogramm unabhängig von den statistischen Daten eine Indexsuche anstelle einer Tabellensuche:

  – Wenn alle Spalten, auf die verwiesen wird, im Index vorhanden sind.
  – Wenn der Index bei der Indexsuche ein Vergleichselement anwenden kann.

Wenn es sich bei der Tabelle um eine typisierte Tabelle handelt, wird die Verwendung der Anweisung ALTER TABLE...VOLATILE nur für die Stammtabelle der Hierarchie der typisierten Tabelle unterstützt.

Zugehörige Konzepte:

- „Vor- und Nachteile von Indizes" auf Seite 288
- „Indexplanung - Tipps" auf Seite 290
- „Indexstruktur" auf Seite 27
- „Indexzugriff und Clusterverhältnisse" auf Seite 180
- „Tabellenreorganisation" auf Seite 280
- „Indexreorganisation" auf Seite 298
- „Tabellen- und Indexverwaltung für Standardtabellen" auf Seite 20
- „Online-Indexdefragmentierung" auf Seite 300
- „Tabellen- und Indexverwaltung für MDC-Tabellen" auf Seite 24
- „Indexbereinigung und Indexpflege" auf Seite 296

Zugehörige Referenzen:

- „intra_parallel - Partitionsinterne Parallelität aktivieren" auf Seite 522

Indexbereinigung und Indexpflege

Wenn Sie Indizes erstellen, sinkt die Leistung, sofern Sie nicht dafür sorgen, dass die Indizes kompakt und organisiert bleiben. Berücksichtigen Sie die folgenden Empfehlungen, um Indizes so klein und effizient wie möglich zu halten:

- Aktivieren Sie die Online-Indexdefragmentierung.

  Erstellen Sie Indizes mit der Klausel MINPCTUSED. Löschen Sie vorhandene Indizes und erstellen Sie sie erneut, falls erforderlich.
- Führen Sie häufige COMMIT-Operationen durch oder aktivieren Sie X-Sperren für Tabellen, entweder explizit oder durch Sperreneskalation, falls häufige COMMIT-Operationen nicht möglich sind.

  Indexschlüssel, die als gelöscht markiert sind, können nach der COMMIT-Operation aus der Tabelle entfernt werden. X-Sperren für Tabellen ermöglichen es, den gelöschten Schlüssel physisch zu entfernen, wenn er als gelöscht markiert ist, wie weiter unten erläutert wird.
- Verwenden Sie REORGCHK, um festzustellen, wann Indizes oder Tabellen (oder beides) zu reorganisieren sind und wann REORG INDEXES mit der Option CLEANUP ONLY zu verwenden ist.

  Um einen Schreib- und Lesezugriff auf einen Index bei der Reorganisation zuzulassen, führen Sie den Befehl REORG INDEXES mit der Option ALLOW WRITE ACCESS aus.

Anmerkung: In DB2® Version 8.1 und späteren Versionen werden alle Indizes als Indizes des Typs 2 erstellt. Eine Ausnahme besteht dann, wenn Sie einer Tabelle einen Index hinzufügen, die bereits Indizes des Typs 1 hat. In diesem Fall wird auch der neue Index als Index des Typs 1 erstellt. Zur Ermittlung des Typs von Index, der für eine Tabelle vorhanden ist, führen Sie den Befehl INSPECT aus. Zur Umwandlung von Indizes des Typs 1 in Indizes des Typs 2 führen Sie den Befehl REORG INDEXES aus.

Die Hauptvorteile von Indizes des Typs 2 lassen sich folgendermaßen beschreiben:

- Ein Index kann für Spalten erstellt werden, deren Länge 255 Byte überschreitet.
- Das Sperren der nächsten Schlüssel wird auf ein Minimum reduziert, wodurch sich der gemeinsame Zugriff verbessert. Der größte Teil des Sperrens der nächsten Schlüssel wird vermieden, weil ein Schlüssel nur als gelöscht markiert, jedoch nicht physisch von der Indexseite entfernt wird. Informationen über das Sperren von Schlüsseln finden Sie in den Abschnitten über die Auswirkungen von Sperren auf die Leistung.

Indexschlüssel, die als gelöscht markiert sind, werden bei folgenden Aktionen bereinigt:

- Während nachfolgender Einfüge-, Aktualisierungs- oder Löschaktivitäten

  Während des Einfügens von Schlüsseln werden Schlüssel, die als gelöscht markiert und bekanntermaßen festgeschrieben sind, bereinigt, wenn eine solche Bereinigung die Durchführung einer Seitentrennung vermeiden hilft und verhindert, dass der Index größer wird.

  Während des Löschens von Schlüsseln wird, wenn sämtliche Schlüssel auf einer Seite als gelöscht markiert wurden, ein Versuch unternommen, eine andere Indexseite zu finden, auf der alle Schlüssel als gelöscht markiert sind und alle diese Löschungen festgeschrieben wurden. Wenn eine solche Seite gefunden wird, wird sie aus der Indexstruktur gelöscht.

  Wenn beim Löschen eines Schlüssels eine X-Sperre für die Tabelle aktiv ist, wird der Schlüssel physisch gelöscht und nicht nur als gelöscht markiert. Während dieser physischen Löschung werden alle gelöschten Schlüssel auf derselben Seite ebenfalls entfernt, wenn sie als gelöscht markiert sind und bekannt ist, dass diese Löschung festgeschrieben ist.
- Bei der Ausführung des Befehls REORG INDEXES mit CLEANUP-Optionen

  Die Option CLEANUP ONLY PAGES sucht nach Indexseiten und gibt diese frei, wenn auf ihnen alle Schlüssel als gelöscht markiert sind und diese Löschung festgeschrieben wurde.

  Die Option CLEANUP ONLY ALL gibt nicht nur Indexseiten frei, auf denen alle Schlüssel als gelöscht markiert und festgeschrieben sind, sondern entfernt zudem auch Satz-IDs (RIDs), die als gelöscht markiert sind und deren Löschung festgeschrieben wurde, auf Seiten, die auch einige nicht gelöschte Satz-IDs enthalten.

  Diese Option versucht außerdem, benachbarte Blattseiten zusammenzufügen, wenn dies zu einer zusammengefügten Blattseite führt, die mindestens den durch PCTFREE angegebenen freien Speicherbereich enthält. Der Wert für PCTFREE ist der Prozentsatz an freiem Speicherbereich, der für den Index bei seiner Erstellung definiert ist. Der Standardwert für PCTFREE ist 10 %. Wenn zwei Seiten zusammengefügt werden können, wird eine der Seiten freigegeben.
- Bei einer Neuerstellung eines Index

  Zu den Dienstprogrammen, die Indizes neu erstellen, gehören folgende:

  – Der Befehl REORG INDEXES, wenn er nicht mit einer der CLEANUP-Optionen verwendet wird
  – Der Befehl REORG TABLE, wenn er nicht mit der Option INPLACE verwendet wird
  – Der Befehl IMPORT mit der Option REPLACE
  – Der Befehl LOAD mit der Option INDEXING MODE REBUILD

Zugehörige Konzepte:
- „Vor- und Nachteile von Indizes" auf Seite 288
- „Indexplanung - Tipps" auf Seite 290
- „Indexstruktur" auf Seite 27
- „Tabellenreorganisation" auf Seite 280
- „Indexreorganisation" auf Seite 298

Indexreorganisation

Wenn Tabellen durch Löschungen und Einfügungen aktualisiert werden, verschlechtert sich die Indexleistung auf folgende Weisen:
- Aufteilung von Blattseiten

  Wenn Blattseiten aufgeteilt (fragmentiert) sind, steigen die E/A-Aufwände, weil mehr Blattseiten gelesen werden müssen, um Tabellenseiten abzurufen.
- Die physische Indexseitenreihenfolge stimmt nicht mehr mit der Reihenfolge der Schlüssel auf diesen Seiten überein. Dies wird als Index mit *schlechter Clusterbildung* bezeichnet.

  Wenn Blattseiten schlechte Clusterbildung aufweisen, ist ein sequenzieller Vorablesezugriff nicht effizient, was zu längeren E/A-Wartezeiten führt.
- Der Index bildet mehr als die maximal effiziente Anzahl von Stufen.

  In einem solchen Fall sollte der Index reorganisiert werden.

Wenn Sie den Parameter MINPCTUSED bei der Erstellung eine Index definieren, führt der Datenbankserver automatisch Indexblattseiten zusammen, wenn ein Schlüssel gelöscht wird und der freie Speicherplatz weniger als der angegebene Prozentsatz ist. Dieser Vorgang wird als *Online-Indexdefragmentierung* bezeichnet. Um die Indexclusterbildung wiederherzustellen, Speicherplatz freizugeben und die Anzahl von Blattseitenstufen zu verringern, können Sie eine der folgenden Methoden anwenden:
- Löschen und erneutes Erstellen des Index
- Reorganisieren von Indizes online mit Hilfe des Befehls REORG INDEXES

  Sie können diese Methode in einer Produktionsumgebung wählen, da sie Benutzern ermöglicht, die Tabelle zu lesen und Daten in sie zu schreiben, während ihre Indizes neu erstellt werden.
- Verwenden des Befehls REORG TABLE mit Optionen, die eine Offlinereorganisation der Tabelle und ihrer Indizes ermöglichen

Online-Indexreorganisation

Wenn Sie den Befehl REORG INDEXES mit der Option ALLOW WRITE ACCESS verwenden, werden alle Indizes für die angegebene Tabelle erneut erstellt, während der Lese- und Schreibzugriff auf die Tabelle möglich ist. Alle Änderungen an der zugrunde liegenden Tabelle, die während des Reorganisationsprozesses einen Einfluss auf die Indizes haben würden, werden in den DB2®-Protokollen aufgezeichnet. Außerdem werden dieselben Änderungen in den internen Speicherpufferbereich gestellt, sofern ein solcher Speicherbereich zur Verfügung steht. Bei der Reorganisation werden die protokollierten Änderungen während des erneuten Erstellens der Indizes zwecks Abgleichs mit der aktuellen Schreibaktivität verarbeitet. Der interne Speicherpufferbereich ist ein ausgewiesener Speicherbereich, der bei Bedarf vom Dienstprogrammzwischenspeicher zugeordnet wird, um die Änderungen an den erstellten bzw. reorganisierten Indizes zu speichern. Durch Verwendung dieses Speicherpufferbereichs können die Änderungen bei der Indexreorganisation verarbeitet werden, indem sie zunächst direkt aus dem Speicher gelesen werden. Gegebenenfalls werden anschließend auch die Protokolle gelesen, jedoch erst zu einem viel späteren Zeitpunkt. Der zugeordnete Speicher wird nach

Beendigung der Reorganisation wieder freigegeben. Nach Beendigung der Reorganisation hat der erneut erstellte Index möglicherweise keine perfekte Clusterbildung. Wenn PCTFREE für einen Index angegeben wird, wird der angegebene Prozentsatz an Speicherplatz auf jeder Seite bei der Reorganisation freigehalten.

Anmerkung: Die Option CLEANUP ONLY des Befehls REORG INDEXES führt keine vollständige Reorganisation der Indizes durch. Die Option CLEANUP ONLY ALL entfernt die Schlüssel, die als gelöscht markiert sind und deren Löschung bekanntermaßen festgeschrieben ist. Sie gibt außerdem Seiten frei, auf denen alle Schlüssel als gelöscht markiert sind und diese Löschung bekanntermaßen festgeschrieben wurde. Bei der Freigabe von Seiten werden benachbarte Blattseiten zusammengefügt, wenn dadurch mindestens der Wert von PCTFREE an freiem Speicherplatz auf der zusammengefügten Seite übrig behalten werden kann. Der Wert für PCTFREE ist der Prozentsatz an freiem Speicherbereich, der für den Index bei seiner Erstellung definiert wird. Die Option CLEANUP ONLY PAGES löscht nur Seiten, auf denen alle Schlüssel als gelöscht markiert sind und diese Löschung bekanntermaßen festgeschrieben ist.

Für den Befehl REORG INDEXES gelten folgende Vorraussetzungen:

- Die Berechtigung SYSADM, SYSMAINT, SYSCTRL oder DBADM oder das Zugriffsrecht CONTROL für die Indizes und die Tabelle
- Eine Menge an freiem Speicherplatz in dem Tabellenbereich, in dem die Indizes gespeichert werden, die der aktuellen Größe des Index entspricht Ziehen Sie in Betracht, die Indizes, die einer Reorganisation unterliegen, in einem großen Tabellenbereich anzulegen, wenn Sie die Anweisung CREATE TABLE ausführen.
- Zusätzlicher Protokollspeicherbereich

  Der Befehl REORG INDEXES protokolliert seine Aktivitäten. Infolgedessen kann die Reorganisation fehlschlagen, insbesondere wenn das System ausgelastet ist und andere gleichzeitige Aktivitäten protokolliert werden.

  Anmerkung: Wenn der Befehl REORG INDEXES ALL mit der Option ALLOW NO ACCESS fehlschlägt, werden die Indizes als beschädigt markiert und die Operation rückgängig gemacht. Wenn jedoch ein REORG-Befehl mit der Option ALLOW READ ACCESS oder ein REORG-Befehl mit der Option ALLOW WRITE ACCESS fehlschlägt, wird das ursprüngliche Indexobjekt wiederhergestellt.

Zugehörige Konzepte:

- „Vor- und Nachteile von Indizes" auf Seite 288
- „Indexplanung - Tipps" auf Seite 290
- „Indexleistung - Tipps" auf Seite 293
- „Online-Indexdefragmentierung" auf Seite 300
- „Indexbereinigung und Indexpflege" auf Seite 296

Zugehörige Tasks:

- „Auswählen einer Methode zur Tabellenreorganisation" auf Seite 286

Online-Indexdefragmentierung

Eine Onlinedefragmentierung wird durch den benutzerdefinierbaren Schwellenwert für die minimale Größe des verwendeten Speicherbereichs auf einer Indexseite (Blattseite) ermöglicht. Wenn ein Indexschlüssel aus einer Blattseite gelöscht und dabei der Schwellenwert überschritten wird, werden die benachbarten Indexseiten daraufhin überprüft, ob zwei Blattseiten zusammengefügt werden können. Wenn auf einer Seite ausreichend Platz zum Zusammenfügen zweier benachbarter Seiten vorhanden ist, erfolgt die Zusammenfügung unverzüglich im Hintergrund.

Eine Onlinedefragmentierung des Index ist nur möglich für Indizes, die in Version 6 und späteren Versionen erstellt wurden. Wenn vorhandene Indizes die Möglichkeit der Onlinezusammenfügung benötigen, müssen sie gelöscht und mit der Klausel `MINPCTUSED` erneut erstellt werden. Setzen Sie `MINPCTUSED` auf einen Wert kleiner als 100. Der empfohlene Wert für `MINPCTUSED` ist kleiner als 50, da der Zweck darin besteht, zwei benachbarte Indexblattseiten zusammenzufügen. Durch den Wert 0 für `MINPCTUSED`, der gleichzeitig der Standardwert ist, wird die Onlinedefragmentierung inaktiviert.

Seiten im Index werden freigegeben, wenn der letzte Indexschlüssel auf der betreffenden Seite entfernt wird. Eine Ausnahme zu dieser Regel ist der Fall, wenn Sie die Klausel `MINPCTUSED` in der Anweisung CREATE INDEX angeben. Mit der Klausel `MINPCTUSED` wird ein Prozentwert des Speicherplatzes auf einer Indexblattseite angegeben. Wenn ein Indexschlüssel gelöscht wird und der Prozentsatz des gefüllten Speicherplatzes auf der Seite bei oder unter dem angegebenen Wert liegt, versucht der Datenbankmanager die verbleibenden Schlüssel mit Schlüsseln einer benachbarten Seite auf eine Seite zusammenzufügen. Wenn genügend Platz auf einer benachbarten Seite vorhanden ist, wird die Zusammenfügung ausgeführt und eine Indexblattseite gelöscht.

Seiten von Indizes, die keine Blattseiten sind, werden bei einer Online-Indexdefragmentierung nicht zusammengefügt. Jedoch werden leere Nichtblattseiten gelöscht und zur Wiederverwendung durch andere Indizes für die gleiche Tabelle verfügbar gemacht. Um diese Nichtblattseiten für andere Objekte in einem DMS-Speichermodell freizugeben oder um Plattenspeicherplatz in einem SMS-Speichermodell freizugeben, führen Sie eine vollständige Reorganisation der Tabelle oder der Indizes aus. Eine vollständige Reorganisation der Tabelle und der Indizes kann den Index auf minimale Größe verkleinern. Nichtblattseiten von Indizes werden bei einer Online-Indexdefragmentierung nicht zusammengefügt, jedoch gelöscht und zur Wiederverwendung freigegeben, wenn sie leer werden. Die Anzahl der Stufen im Index und die Anzahl der Blatt- und Nichtblattseiten können sich verringern.

Für Indizes des Typs 2 werden Schlüssel von einer Seite beim Löschen von Schlüsseln nur dann entfernt, wenn eine exklusive Sperre (Modus X) für die Tabelle aktiviert ist. Bei einer solchen Operation ist eine Online-Indexdefragmentierung effektiv. Wenn beim Löschen von Schlüsseln hingegen keine X-Sperre für die Tabelle aktiviert ist, werden Schlüssel als gelöscht markiert, jedoch nicht physisch von der Indexseite entfernt. Infolgedessen wird auch kein Versuch einer Defragmentierung unternommen.

Zur Defragmentierung von Indizes des Typs 2, in denen Schlüssel zwar als gelöscht markiert werden, jedoch in der physischen Indexseite verbleiben, führen Sie den Befehl REORG INDEXES mit der Option CLEANUP ONLY ALL aus. Die Option CLEANUP ONLY ALL defragmentiert den Index ohne Rücksicht auf den Wert von MINPCTUSED. Wenn Sie den Befehl REORG INDEXES mit der Option CLEANUP ONLY ALL ausführen, werden zwei benachbarte Blattseiten zusammen-

gefügt, wenn eine solche Zusammenfügung mindestens den Wert von PCTFREE an freiem Speicherplatz auf der zusammengefügten Seite übrig lassen kann. PCTFREE wird bei der Erstellung eines Index angegeben und hat den Standardwert 10 Prozent.

Zugehörige Konzepte:

- „Vor- und Nachteile von Indizes" auf Seite 288
- „Indexleistung - Tipps" auf Seite 293
- „Indexstruktur" auf Seite 27
- „Indexreorganisation" auf Seite 298

Informationen zu DMS-Einheiten

Wenn Sie vom Datenbankmanager verwaltete DMS-Einheitenbehälter (DMS - Database Managed Storage) für Tabellenbereiche verwenden, beachten Sie die folgenden Gesichtspunkte für eine effektive Verwaltung:

- **Zwischenspeichern von Dateisystemen im Cache**

  Das Zwischenspeichern von Dateisystemen wird wie folgt ausgeführt:

  – Seiten aus DMS-Dateibehältern (und aus allen SMS-Behältern) können vom Betriebssystem im Dateisystemcache zwischengespeichert werden.

  – Seiten aus den Tabellenbereichen von DMS-Einheitenbehältern werden vom Betriebssystem nicht im Dateisystemcache zwischengespeichert.

  Anmerkung: Unter Windows® NT gibt die Registriervariable DB2NTNOCACHE an, ob DB2® Datenbankdateien mit der Option NOCACHE öffnet oder nicht. Wenn DB2NTNOCACHE=ON definiert ist, wird das Zwischenspeichern im Systemcache inaktiviert. Wenn DB2NTNOCACHE=OFF definiert 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, so dass der Pufferpool oder der Sortierspeicher vergrößert werden kann.

- **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. Für Tabellenbereiche, die mit SMS- oder DMS-Dateibehältern arbeiten, kann die oben beschriebene Cachefunktion des Dateisystems die E/A-Operationen überflüssig machen, die in einem solchen Fall ansonsten anfallen.

  Tabellenbereiche, die DMS-Einheitenbehälter 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 Einheitenbehälter 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.

- **Verwenden von LOB- oder LONG-Daten**

  Wenn eine Anwendung LOB- oder LONG-Daten abruft, verwendet der Datenbankmanager keine seiner Puffer, um die Daten zwischenzuspeichern. Jedes Mal, wenn eine Anwendung eine dieser Seiten benötigt, muss der Datenbankmanager sie vom Plattenspeicher abrufen. Wenn LOB- oder LONG-Daten jedoch in SMS- oder DMS-Dateibehältern gespeichert werden, kann die Pufferung durch den Dateisystemcache erfolgen und dadurch eine bessere Leistung erreicht werden.

  Da die Systemkataloge einige LOB-Spalten enthalten, sollten Sie sie in SMS-Tabellenbereichen oder in DMS-Dateitabellenbereichen speichern.

Zugehörige Konzepte:

- „Verzeichnisse und Dateien einer Datenbank" auf Seite 13
- „SMS-Tabellenbereiche" auf Seite 16
- „DMS-Tabellenbereiche" auf Seite 17

Agentenverwaltung

Dieser Abschnitt beschreibt die Verwendung von Agenten durch den Datenbankmanager und erläutert, wie Agenten im Hinblick auf optimale Leistung verwaltet werden.

Datenbankagenten

Für jede Datenbank, auf die eine Anwendung zugreift, werden verschiedene Prozesse oder Threads gestartet, um die verschiedenen Anwendungstasks durchzuführen. Zu diesen Tasks gehören das Protokollieren, die Kommunikation und der Vorablesezugriff.

Datenbankagenten sind EDU-Prozesse oder EDU-Threads (EDU - Engine Dispatchable Unit, zuteilbare Einheit der Steuerkomponente). Datenbankagenten erledigen die Arbeit im Datenbankmanager, die von Anwendungen angefordert wird. In UNIX®-Umgebungen werden diese Agenten als Prozesse ausgeführt. Unter Intel-basierten Betriebssystemen wie Windows® werden die Agenten als Threads ausgeführt.

Die maximale Anzahl von Anwendungsverbindungen wird durch den Konfigurationsparameter *max_connections* des Datenbankmanagers gesteuert. Die Arbeit jeder Anwendungsverbindung wird durch einen einzigen Verarbeitungsagenten koordiniert.

Ein *Verarbeitungsagent* führt Anwendungsanforderungen aus, ist aber nicht dauerhaft mit einer bestimmten Anwendung verbunden. Der Koordinatoragent verfügt über alle Informationen und Steuerblöcke, die zur Durchführung von Aktionen im Datenbankmanager erforderlich sind, die von der Anwendung angefordert wurden.

Es gibt vier Typen von Verarbeitungsagenten:

- Nicht zugeordnete Agenten
- Inaktive Agenten
- Aktive Koordinatoragenten
- Subagenten

Nicht zugeordnete Agenten
Dies ist die einfachste Form eines Verarbeitungsagenten. Er hat keine abgehende Verbindung und keine lokale Datenbankverbindung oder Exemplarverbindung.

Inaktive Agenten
Ein inaktiver Agent ist ein Verarbeitungsagent, der sich nicht in einer aktiven Transaktion befindet, keine abgehende Verbindung hat und keine lokale Datenbank- oder Exemplarverbindung hat. Inaktive Agenten sind verfügbar, um Arbeit für eine Anwendungsverbindung zu übernehmen.

Aktive Koordinatoragenten
Jeder Prozess oder Thread einer Clientanwendung hat einen einzelnen aktiven Agenten, der die Arbeit der Anwendung an der Datenbank koordiniert. Wenn der Koordinatoragent erstellt ist, führt er alle Datenbankanforderungen für die zugehörige Anwendung aus und kommuniziert mit anderen Agenten per Interprozesskommunikation (IPC) oder über Protokolle zur Fernverbindung. Jeder Agent arbeitet mit seinem eigenen privaten Speicher und benutzt Ressourcen des Datenbankmanagers und globale Datenbankressourcen, wie z. B. den Pufferpool, gemeinsam mit anderen Agenten. Wenn eine Transaktion abgeschlossen wird, kann der aktive Koordinatoragent zu einem inaktiven Agenten werden.

Wenn ein Client die Verbindung zu einer Datenbank oder zu einem Exemplar trennt, geschieht mit dem koordinierenden Agenten Folgendes:

- Er wird ein aktiver Agent. Wenn andere Verbindungen warten, wird der Verarbeitungsagent ein aktiver Koordinatoragent.
- Er wird freigegeben und als nicht zugeordnet markiert, wenn keine Verbindungen warten und die maximale Zahl von Poolagenten noch nicht erreicht wurde.
- Er wird beendet und der von ihm verwendete Speicher wird freigegeben, wenn keine Verbindungen warten und die maximale Zahl von Poolagenten erreicht wurde.

Subagenten
In Umgebungen mit partitionierten Datenbanken und Umgebungen mit aktivierter partitionsinterner Parallelität verteilt der Koordinatoragent die Datenbankanforderungen an Subagenten, die ihrerseits die Anforderungen für die Anwendung ausführen. Wenn der Koordinatoragent erstellt ist, verarbeitet er alle Datenbankanforderungen für seine Anwendung, indem er die Subagenten koordiniert, die die Anforderungen in der Datenbank ausführen.

Agenten, die keine Arbeit für Anwendungen ausführen und darauf warten, zugeordnet zu werden, gelten als nicht zugeordnete Agenten im Bereitschaftsmodus und befinden sich in einem *Agentenpool*. Diese Agenten sind für Anforderungen von Koordinatoragenten, die für Clientprogramme aktiv sind, oder für Subagenten, die für vorhandene Koordinatoragenten aktiv sind, verfügbar. Die Anzahl der verfügbaren Agenten hängt vom Wert der Konfigurationsparameter *maxagents* und *num_poolagents* des Datenbankmanagers ab.

Wenn ein Agent seine Arbeit beendet, jedoch noch eine Verbindung zu einer Datenbank hat, wird er in den Agentenpool versetzt. Ungeachtet dessen, ob der Verbindungskonzentrator für die Datenbank aktiviert ist, wird ein Agent beendet, wenn er nicht innerhalb eines bestimmten Zeitraums zur Bedienung einer neuen Anforderung aktiviert wird oder wenn die aktuelle Anzahl aktiver und im Pool befindlicher Agenten größer als der Wert von *num_poolagents* ist.

Agenten aus dem Agentenpool (*num_poolagents*) werden als Koordinatoragenten für die folgenden Arten von Anwendungen wiederverwendet:

- Ferne TCP/IP-basierte Anwendungen
- Lokale Anwendungen auf UNIX-Betriebssystemen
- Lokale und ferne Anwendungen auf Windows-Betriebssystemen

Andere Arten ferner Anwendungen erstellen stets einen neuen Agenten. Falls kein Agent im Bereitschaftsmodus vorhanden ist, wenn ein Agent angefordert wird, wird ein neuer Agent dynamisch erstellt. Da die Erstellung eines neuen Agenten einen bestimmten Systemaufwand erfordert, ist die CONNECT- und ATTACH-Leistung besser, wenn ein nicht zugeordneter Agent im Bereitschaftsmodus für einen Client aktiviert werden kann.

Wenn ein Subagent für eine Anwendung aktiv ist, wird er dieser Anwendung *zugeordnet*. Nach Beendigung der angeforderten Operationen kann er in den Agentenpool versetzt werden, bleibt jedoch der ursprünglichen Anwendung zugeordnet. Wenn die Anwendung eine weitere Operation anfordert, überprüft der Datenbankmanager zunächst den Agentenpool nach zugeordneten inaktiven Agenten, bevor er einen neuen Agenten erstellt.

Zugehörige Konzepte:

- „Verwaltung von Datenbankagenten" auf Seite 304
- „Agenten in einer partitionierten Datenbank" auf Seite 308
- „Verbesserungen des Verbindungskonzentrators für Clientverbindungen" auf Seite 306
- „Konfigurationsparameter mit Auswirkung auf die Anzahl von Agenten" auf Seite 305

Verwaltung von Datenbankagenten

Die meisten Anwendungen richten eine Eins-zu-eins-Beziehung zwischen der Zahl der verbundenen Anwendungen und der Zahl der Anwendungsanforderungen ein, die von der Datenbank verarbeitet werden können. Es kann jedoch sein, dass Sie aufgrund Ihrer Arbeitsumgebung eine Viele-zu-eins-Beziehung zwischen der Zahl der verbundenen Anwendungen und der Zahl der Anwendungsanforderungen benötigen, die verarbeitet werden können.

Die Möglichkeit, diese Faktoren separat zu steuern, wird durch Konfigurationsparameter des Datenbankmanagers bereitgestellt:

- Der Parameter *max_connections* gibt die Anzahl verbundener Anwendungen an.
- Der Parameter *max_coordagents* gibt die Anzahl von Anwendungsanforderungen an, die verarbeitet werden können.

Der Verbindungskonzentrator ist aktiviert, wenn der Wert des Parameters *max_connections* größer als der Wert des Parameters *max_coordagents* ist.

Da jeder aktive Koordinatoragent globale Ressourcen erfordert, steigt mit wachsender Zahl dieser Agenten auch die Wahrscheinlichkeit, dass die Obergrenze an verfügbaren globalen Datenbankressourcen erreicht wird. Um das Erreichen der Obergrenze der verfügbaren globalen Datenbankressourcen zu vermeiden, können Sie den Parameter *max_connections* auf einen höheren Wert als den Parameter *max_coordagents* setzen.

Zugehörige Konzepte:

- „Agenten in einer partitionierten Datenbank" auf Seite 308
- „Verbesserungen des Verbindungskonzentrators für Clientverbindungen" auf Seite 306
- „Konfigurationsparameter mit Auswirkung auf die Anzahl von Agenten" auf Seite 305

Konfigurationsparameter mit Auswirkung auf die Anzahl von Agenten

Die folgenden Konfigurationsparameter des Datenbankmanagers legen fest, wie viele Datenbankagenten erstellt werden und wie sie verwaltet werden:

- Maximale Anzahl von Agenten (*maxagents*): Die Anzahl von Agenten, die gleichzeitig aktiv sein können. Dieser Wert bezieht sich auf die Gesamtanzahl von Agenten, die für alle Anwendungen aktiv sind, einschließlich Koordinatoragenten, Subagenten, inaktive Agenten und Agenten im Bereitschaftsmodus.
- Agentenpoolgröße (*num_poolagents*): Die Gesamtanzahl von Agenten, einschließlich aktiver Agenten und Agenten im Agentenpool, die im System verfügbar gehalten werden. Der Standardwert für diesen Parameter ist die Hälfte der durch *maxagents* angegebenen Anzahl.
- 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 Partition zulässig sind.
- Maximale Anzahl koordinierender Agenten (*max_coordagents*): In Umgebungen mit partitionierten Datenbanken und Umgebungen mit aktivierter partitionsinterner Parallelität und aktiviertem Verbindungskoordinator begrenzt dieser Wert die Anzahl koordinierender Agenten.
- Maximale Anzahl gleichzeitig aktiver Agenten (*maxcagents*): Mit diesem Wert wird die Anzahl von *Token* gesteuert, die der Datenbankmanager zulässt. Für jede Datenbanktransaktion (Arbeitseinheit), die stattfindet, wenn ein Client mit einer Datenbank verbunden ist, muss ein koordinierender Agent die Berechtigung zur Verarbeitung vom Datenbankmanager einholen. Diese Berechtigung wird als *Verarbeitungstoken* bezeichnet. Der Datenbankmanager erlaubt nur Agenten mit einem Verarbeitungstoken, eine Arbeitseinheit an einer Datenbank auszuführen. Wenn kein Token verfügbar ist, muss der Agent warten, bis ein Token zur Verarbeitung der Transaktion verfügbar ist.

  Dieser Parameter kann in Umgebungen nützlich sein, in denen der Leistungsbedarf in Zeiten hoher Auslastung die Kapazität der vorhandenen Systemressourcen für Hauptspeicher, CPU und Plattenspeicher übersteigt. In einer solchen Umgebung könnte zum Beispiel das Aus- und Einlagern von Seiten (Paging) eine Leistungsbeeinträchtigung in Spitzenlastzeiten zur Folge haben. Mit diesem Parameter können Sie die Auslastung steuern und den Leistungsabfall verhindern. Er kann sich jedoch auf den gleichzeitigen Zugriff, auf Wartezeiten oder auf beides auswirken.

Zugehörige Konzepte:

- „Datenbankagenten" auf Seite 302
- „Verwaltung von Datenbankagenten" auf Seite 304
- „Agenten in einer partitionierten Datenbank" auf Seite 308

Verbesserungen des Verbindungskonzentrators für Clientverbindungen

Für Internetanwendungen mit vielen relativ kurzfristigen Verbindungen oder ähnliche Arten von Anwendungen verbessert der Verbindungskonzentrator die Leistung, indem er es ermöglicht, wesentlich mehr Clientverbindungen effizient zu verarbeiten. Darüber hinaus verringert er die Speicherauslastung für jede Verbindung und senkt die Anzahl von Kontextumschaltungen.

Anmerkung: Der Verbindungskonzentrator ist aktiviert, wenn der Wert des Parameters *max_connections* größer als der Wert des Parameters *max_coordagents* ist.

In einer Umgebung, die viele gleichzeitige Benutzerverbindungen erfordert, können Sie den Verbindungskonzentrator aktivieren, um die Systemressourcen effizienter zu nutzen. Diese Einrichtung bietet Vorteile, die früher nur im Verbindungspooling von DB2 Connect zu finden waren. Sowohl das Verbindungspooling als auch der Verbindungskonzentrator werden im DB2 Connect Benutzerhandbuch beschrieben. Nach Herstellung der ersten Verbindung verringert der Verbindungskonzentrator die Zeit für die Verbindung zu einem Host. Wenn die Trennung einer Verbindung zu einem Host angefordert wird, wird die eingehende Verbindung gelöscht, während die abgehende Verbindung zum Host in einem Pool erhalten bleibt. Wenn eine neue Anforderung zur Herstellung einer Verbindung zu dem Host erfolgt, versucht DB2® eine vorhandene abgehende Verbindung aus dem Pool wiederzuverwenden.

Anmerkung: Wenn Anwendungen mit einem Verbindungspooling oder mit dem Verbindungskonzentrator arbeiten, erreichen Sie die beste Leistung, indem Sie die Parameter optimieren, die die Größe des Datenblocks steuern, der im Cache zwischengespeichert wird. Weitere Informationen finden Sie im DB2 Connect Benutzerhandbuch.

Mit DB2 Connect-Verbindungspools und dem Verbindungskonzentrator schließt der aktive Agent seine abgehende Verbindung nicht, nachdem der Client seine Verbindung getrennt hat, sondern er wird in den Agentenpool für die Anwendung versetzt, wo er zu einem *logischen Subagenten* wird, der von einem *logischen Koordinatoragenten* mit einer aktiven Verbindung zu dem fernen Host gesteuert wird.

Beim Einsatz der Poolfunktion für Verbindungen ist DB2 Connect™ auf eingehende TCP/IP- und abgehende TCP/IP- und SNA-Verbindungen beschränkt. Bei Verwendung von SNA muss der Sicherheitstyp auf den Wert `NONE` (Keine) eingestellt sein, damit die Verbindung in den Pool gesetzt werden kann. Bei Verbindungspools werden diese Agenten im Bereitschaftsmodus als *inaktive Agenten* bezeichnet. Die Bezeichnung „Pool für inaktive Agenten" ist mit „Pool für abgehende Verbindungen" synonym. Der Verbindungskonzentrator implementiert eine ähnliche Methode zur Beibehaltung inaktiver Agenten zur späteren Verwendung in einem anwendungsspezifischen Pool.

Verwendungsbeispiele:

1. Betrachten Sie eine ESE-Umgebung mit einer einzelnen Datenbankpartition, in der durchschnittlich 1000 Benutzer mit der Datenbank verbunden sind. Manchmal steigt die Anzahl gleichzeitiger Transaktion bis auf 200 an, wird jedoch nie höher als 250. Transaktionen sind kurz.

Für diese Auslastung definiert der Administrator die folgenden Konfigurationsparameter des Datenbankmanagers:

- *max_connections* wird auf 1000 gesetzt, um eine Unterstützung der durchschnittlichen Anzahl von Verbindungen sicherzustellen.
- *max_coordagents* wird auf 250 gesetzt, um die maximale Anzahl gleichzeitiger Transaktionen zu unterstützen.
- *maxagents* wird mit einem ausreichend hohen Wert definiert, um alle Koordinatoragenten und Subagenten (wenn zutreffend) zu unterstützen, die zur Ausführung von Transaktionen auf dem Knoten erforderlich sind.

  Wenn *intra_parallel* den Wert OFF hat, wird *maxagents* auf den Wert 250 gesetzt, weil in einer solchen Umgebung keine Subagenten verfügbar sind. Wenn *intra_parallel* den Wert ON hat, sollte *maxagents* auf einen ausreichend hohen Wert gesetzt werden, um den Koordinatoragenten und die Subagenten zu berücksichtigen, die für jede Transaktion erforderlich sind, die auf Daten auf dem Knoten zugreift. Wenn zum Beispiel jede Transaktion vier Subagenten erfordert, sollte *maxagents* auf den Wert (4+1) * 250, also 1250, gesetzt werden. Zur weiteren Optimierung des Parameters *maxagents* erfassen Sie Monitormomentaufnahmen für den Datenbankmanager. Die erreichte obere Grenze für die Anzahl der Agenten gibt die geeignete Einstellung für *maxagents* an.
- *num_poolagents* wird abhängig vom Wert für *maxagents* auf einen Wert von mindestens 250 bzw. einen Wert bis maximal 1250 gesetzt, um sicherzustellen, dass genügend Datenbankagenten verfügbar sind, um ankommende Clientanforderungen ohne Aufwand zur Erstellung neuer Agenten verarbeiten zu können.

  Allerdings sollte diese Anzahl gesenkt werden, um die Ressourcennutzung in Zeiten geringer Auslastung zu reduzieren. Wenn dieser Wert zu niedrig eingestellt wird, werden Agenten beendet anstatt in den Agentenpool versetzt zu werden, was eine Erstellung neuer Agenten erforderlich macht, bevor der Server eine durchschnittliche Auslastung verarbeiten kann.
- *num_init_agents* wird auf den gleichen Wert wie *num_poolagents* gesetzt, da die Anzahl der Agenten bekannt ist, die aktiv sein sollten. Dies veranlasst den Datenbankmanager, eine angemessene Anzahl von Agenten beim Starten zu erstellen, anstatt sie vor der Verarbeitung einer Anforderung zu erstellen.

Die Fähigkeit der zugrunde liegenden Hardware, eine bestimmte Auslastung zu bewältigen, wird an dieser Stelle nicht behandelt. Wenn die zugrunde liegende Hardware keine Anzahl X von Agenten gleichzeitig verarbeiten kann, müssen Sie diese Anzahl auf den Maximalwert verringern, der von der Hardware unterstützt wird. Wenn zum Beispiel der Maximalwert bei 1500 Agenten liegt, begrenzt dieser Wert die maximale Anzahl gleichzeitiger Transaktionen, die verarbeitet werden können. Sie sollten diese Art leistungsrelevanter Einstellungen überwachen, da es nicht immer möglich ist, die exakte Anzahl von Anforderungen zu bestimmen, die zu einem gegebenen Zeitpunkt an andere Knoten gesendet werden.

2. In einem System, in dem die Auslastung auf ein Maximum von 100 gleichzeitigen Transaktionen und die gleiche Anzahl verbundener Benutzer wie in Beispiel 1 begrenzt werden muss, können Sie die Konfigurationsparameter des Datenbankmanagers wie folgt einstellen:
   - *max_coordagents* auf 100

- *num_poolagents* auf 100

Mit diesen Einstellungen liegt die maximale Anzahl von Clients, die gleichzeitig Transaktionen ausführen können, bei 100. Wenn alle Clients ihre Verbindungen trennen, warten 100 Agenten darauf, neue Clientverbindungen zu bedienen. Sie sollten den Parameter *maxagents* unter Berücksichtigung der Art der Auslastung, der Einstellungen für die abfrageinterne Parallelität, der Anzahl der Datenbankpartitionen und der zugrunde liegenden Hardware einstellen.

3. Betrachten Sie als Nächstes eine ESE-Installation mit fünf Datenbankpartitionen, in der jede Partition eine durchschnittliche Anzahl von 1000 Benutzerverbindungen hat und die gleichzeitigen Transaktionen eine Anzahl von 200, jedoch nie über 250 erreichen. Die Konfigurationsparameter sollten wie folgt definiert werden:
   - *max_coordagents* wird auf 250 gesetzt, weil wie in Beispiel 1 höchstens 250 Clients gleichzeitig Transaktionen ausführen.
   - *maxagents* wird auf 1500 gesetzt. Unter der Annahme, dass die Daten auf fünf Partitionen verteilt sind, könnte jede Transaktion mindestens einen Teilbereich (Subsection) auf jedem Knoten im System ausführen. ((1 Koordinatoragent + 5 Subagenten) * 250 = 1500.)
   - *num_poolagents* wird auf 1200 gesetzt. (Unter der Annahme einer Durchschnittsanzahl von 200 gleichzeitigen Transaktionen mit einem Maximum von 250 ergibt sich als Anzahl von Agenten, die durchschnittlich erforderlich sind: (1+5)*200 = 1200.)
   - *num_init_agents* wird wie in Beispiel 1 auf den gleichen Wert wie *num_poolagents* gesetzt.
4. In einem System, für das Sie den Verbindungskonzentrator nicht aktivieren wollen, jedoch 250 gleichzeitig verbundene Benutzer zulassen wollen, stellen Sie die Konfigurationsparameter des Datenbankmanagers wie folgt ein:
   - *max_connections* auf 250
   - *max_coordagents* auf 250

Zugehörige Konzepte:

- „Datenbankagenten" auf Seite 302
- „Verwaltung von Datenbankagenten" auf Seite 304
- „Übersicht über die Architektur und Prozesse von DB2" auf Seite 9
- „Speicherverwaltung" auf Seite 38

Agenten in einer partitionierten Datenbank

In Umgebungen mit partitionierten Datenbanken und Umgebungen mit aktivierter partitionsinterner Parallelität besitzt jede Partition (d. h. jeder Datenbankserver oder Knoten) einen eigenen Pool von Agenten, aus dem Subagenten entnommen werden. Durch die Verwendung dieses Pools müssen Subagenten nicht jedes Mal erstellt und wieder gelöscht werden, wenn ein Subagent benötigt wird oder seine Arbeit beendet hat. Die Subagenten können als zugeordnete Agenten im Pool bleiben und vom Datenbankmanager für neue Anforderungen von der Anwendung, der sie zugeordnet sind, verwendet werden.

Anmerkung: Wenn der Verbindungskonzentrator aktiviert ist, werden Subagenten nicht unbedingt einer Anwendung zugeordnet.

In Umgebungen mit partitionierten Datenbanken und Umgebungen mit aktivierter partitionsinterner Parallelität besteht eine enge Beziehung zwischen dem Einfluss auf die Leistung und den Speicherbedarf innerhalb des Systems und der Optimierung des Agentenpools:

- Der Konfigurationsparameter des Datenbankmanagers für die Größe des Agentenpools (*num_poolagents*) betrifft die Gesamtanzahl von Subagenten, die Anwendungen in einer Partition (auch als Knoten bezeichnet) zugeordnet bleiben können. Wenn die Poolgröße zu klein ist und der Pool voll ist, wird ein Subagent aus der Zuordnung mit der Anwendung, für die er aktiv ist, gelöst und beendet. Da Subagenten ständig erstellt und Anwendungen erneut zugeordnet werden müssen, sinkt die Leistung.

  Außerdem könnte bereits eine Anwendung den Pool mit zugeordneten Subagenten füllen, wenn der Wert für den Parameter *num_poolagents* zu klein ist. Wenn nun eine andere Anwendung einen neuen Subagenten benötigt und über keine Agenten im zugeordneten Pool verfügt, „stiehlt" sie Subagenten aus den Agentenpools anderer Anwendungen. In diesem Fall ist der Systemaufwand hoch und die Leistung gering.
- Wägen Sie die Vor- und Nachteile zuweniger Agenten gegen den Ressourcenbedarf zu vieler Agenten ab, die zu einem gegebenen Zeitpunkt aktiv sein können.

  Wenn der Wert des Parameters *num_poolagents* zum Beispiel zu groß ist, werden zugeordnete Subagenten lange Zeit ungenutzt im Pool behalten und belegen Datenbankmanagerressourcen, die für andere Tasks nicht verfügbar sind.

  Anmerkung: Wenn der Verbindungskonzentrator aktiviert ist, hat die Anzahl von Agenten, die durch den Parameter *num_poolagents* angegeben wird, nur empfehlenden Charakter. Es können sich zu einem gegebenen Zeitpunkt auch mehr Agenten im Agentenpool befinden.

Andere asynchrone Prozesse und Threads

Neben den Datenbankagenten werden auch noch andere asynchrone Aktivitäten des Datenbankmanagers als eigene Prozesse bzw. Threads ausgeführt. Dazu gehören:

- E/A-Server oder E/A-Vorablesefunktionen der Datenbank
- Asynchrone Seitenlöschfunktionen der Datenbank
- Protokollfunktionen der Datenbank
- Detektoren für gegenseitige Sperren in der Datenbank
- Ereignismonitore
- Übertragungs- und IPC-Empfangsprozesse (Listeners)
- Funktionen zum Neuausgleich von Daten in Tabellenbereichsbehältern

Zugehörige Konzepte:

- „Konfiguration von E/A-Servern für Vorablesezugriff und Parallelität" auf Seite 273
- „Illustration des Vorablesens mit paralleler E/A" auf Seite 275
- „Datenbankagenten" auf Seite 302
- „Verwaltung von Datenbankagenten" auf Seite 304
- „Konfigurationsparameter mit Auswirkung auf die Anzahl von Agenten" auf Seite 305

Informationen des Datenbanksystemmonitors

Der DB2®-Datenbankmanager pflegt Daten über den Betrieb, die Leistung und die Anwendungen, die ihn verwenden. Diese Daten werden während des Betriebs des Datenbankmanagers verwaltet und können wichtige Informationen über die Leistung und zur Fehlerbehebung liefern. Zum Beispiel erhalten Sie Anhaltspunkte über:

- Die Anzahl von Anwendungen, die mit einer Datenbank verbunden sind, ihren Status und welche SQL-Anweisungen (falls überhaupt) von jeder Anwendung ausgeführt werden.
- Informationen, die zeigen, wie gut der Datenbankmanager und die Datenbank konfiguriert sind, und bei der Optimierung helfen.
- Aufgetretene gegenseitige Sperren für eine angegebene Datenbank, beteiligte Anwendungen und welche Sperren in der Konkurrenzsituation waren.
- Die Liste von Sperren, die von einer Anwendungen oder Datenbank aktiviert wurden. Wenn die Anwendung die Verarbeitung nicht fortsetzen kann, weil sie auf eine Sperre wartet, werden weitere Informationen zur Sperre, einschließlich der Anwendung, die sie aktiviert hat, bereitgestellt.

Da die Erfassung einiger dieser Daten mit Systemaufwand für den Betrieb von DB2 verbunden ist, sind **Monitorschalter** verfügbar, mit denen gesteuert werden kann, welche Informationen erfasst werden. Zum expliziten Einstellen der Monitorschalter wird der Befehl UPDATE MONITOR SWITCHES oder die API sqlmon() verwendet. (Sie benötigen hierzu die Berechtigung SYSADM, SYSCTRL oder SYSMAINT.)

Sie können auf die vom Datenbankmanager verwalteten Daten zugreifen, indem Sie entweder eine Momentaufnahme (Snapshot) erstellen oder einen Ereignismonitor verwenden.

Erstellen einer Momentaufnahme

Sie können eine Momentaufnahme auf die zwei folgenden Arten erstellen:

- Verwenden des Befehls GET SNAPSHOT über die Befehlszeile
- Schreiben einer eigenen Anwendung mit einem API-Aufruf sqlmonss()

Verwenden eines Ereignismonitors

Ein Ereignismonitor (Event Monitor) erfasst Systemmonitordaten nach Eintreten bestimmter Ereignisse, wie zum Beispiel das Ende einer Transaktion, das Ende einer Anweisung oder die Erkennung einer gegenseitigen Sperre. Diese Informationen können in Dateien oder eine benannte Pipe geschrieben werden.

Ein Ereignismonitor wird wie folgt verwendet:

1. Erstellen Sie die Definition des Ereignismonitors mit Hilfe der Steuerzentrale oder der SQL-Anweisung CREATE EVENT MONITOR. Diese Anweisung speichert die Definition in den Systemkatalogen der Datenbank.

2. Aktivieren Sie den Ereignismonitor über die Steuerzentrale oder mit der folgenden SQL-Anweisung:

   ```
   SET EVENT MONITOR ergname STATE 1
   ```

   Wenn die Ergebnisse in eine benannte Pipe geschrieben werden, starten Sie die Anwendung, die die Daten aus der benannten Pipe liest, bevor Sie den Ereignismonitor aktivieren. Sie können entweder eine eigene Anwendung zum Lesen schreiben oder den Befehl **db2evmon** verwenden. Sobald der Ereignismonitor aktiv ist und mit dem Schreiben von Ereignissen in die Pipe beginnt, liest **db2evmon** die Ereignisse, wie sie generiert werden, und schreibt Sie in die Standardausgabe.
3. Lesen Sie die Tracedaten. Wenn Sie einen Ereignismonitor verwenden, der in eine Datei schreibt, können Sie die binären Tracedaten, die er erstellt, mit einer der folgenden Methoden anzeigen:
   - Verwenden Sie das Tool **db2evmon** zum Formatieren der Tracedaten für die Standardausgabe.
   - Klicken Sie das Symbol **Event Analyzer** in der Steuerzentrale unter einem Windows®-Betriebssystem an, um eine grafische Schnittstelle zum Anzeigen der Tracedaten, zum Suchen nach Schlüsselwörtern und zum Herausfiltern nicht erwünschter Daten aufzurufen.

     Anmerkung: Wenn das Datenbanksystem, das Sie überwachen, nicht auf derselben Maschine wie die Steuerzentrale ausgeführt wird, müssen Sie die Ereignismonitordatei auf dieselbe Maschine wie die Steuerzentrale kopieren, damit die Tracedaten angezeigt werden können. Sie können die Datei alternativ auch in ein gemeinsam benutztes Dateisystem stellen, auf das beide Maschinen zugreifen können.

Zugehörige Konzepte:

• „Einstiegstipps für die Leistungsoptimierung” auf Seite 7

Kapitel 9. Verwenden von Governor

Dieses Kapitel beschreibt die Einrichtung und Verwendung des Tools Governor zur Überwachung und Steuerung von Datenbankaktivitäten.

Das Dienstprogramm Governor

Das Dienstprogramm Governor kann das Verhalten von Anwendungen überwachen, die an der Datenbank ausgeführt werden, und abhängig von den Regeln, die Sie in der Konfigurationsdatei von Governor angeben, bestimmte Funktionsweisen ändern.

Ein Governor-Exemplar besteht aus einem Front-End-Dienstprogramm und mindestens einem Dämon. Jedes Governor-Exemplar, das Sie starten, ist für ein Exemplar des Datenbankmanagers spezifisch. Wenn Sie das Dienstprogramm Governor starten, wird standardmäßig ein Governor-Dämon in jeder Partition einer partitionierten Datenbank gestartet. Sie können jedoch angeben, dass ein Dämon in einer einzelnen Partition gestartet werden soll, die Sie überwachen wollen.

Anmerkung: Wenn Governor aktiv ist, können die Momentaufnahmenanforderungen des Dienstprogramms die Leistung des Datenbankmanagers beeinträchtigen. Zur Verbesserung der Leistung können Sie das Aktivierungsintervall (wake-up) von Governor vergrößern, um den CPU-Bedarf des Dienstprogramms zu verringern.

Jeder Governor-Dämon sammelt Informationen über die Anwendungen, die an einer Datenbank ausgeführt werden. Anschließend prüft er diese Informationen anhand der Regeln, die Sie in der Konfigurationsdatei von Governor für diese Datenbank angeben.

Governor verwaltet Anwendungstransaktionen, wie es durch die Regeln in der Konfigurationsdatei angegeben ist. Zum Beispiel könnte die Anwendung einer Regel darauf hinweisen, dass eine Anwendung eine bestimmte Ressource übermäßig beansprucht. Eine Regel gibt die durchzuführende Aktion an, wie zum Beispiel das Ändern der Priorität der Anwendung oder das zwangsweise Trennen der Anwendung von der Datenbank.

Wenn die einer Regel zugeordnete Aktion die Priorität der Anwendung ändert, ändert Governor die Priorität von Agenten in der Datenbankpartition, in der die Ressourcenverletzung aufgetreten ist. Wenn in einer partitionierten Datenbank die Anwendung zwangsweise von der Datenbank getrennt wird, findet die Aktion auch dann statt, wenn der Dämon, der die Verletzung festgestellt hat, auf dem Koordinatorknoten der Anwendung ausgeführt wird.

Governor protokolliert alle von ihm durchgeführten Aktionen. Zur Prüfung der Aktionen können Sie die Protokolldateien abfragen.

Zugehörige Konzepte:

- „Der Governor-Dämon" auf Seite 315
- „Die Konfigurationsdatei von Governor" auf Seite 317
- „Governor-Protokolldateien" auf Seite 326

Zugehörige Tasks:

- „Starten und Stoppen von Governor" auf Seite 314
- „Konfigurieren von Governor" auf Seite 316

Zugehörige Referenzen:

- „db2gov - DB2 Governor Command" in *Command Reference*

Starten und Herunterfahren von Governor

Dieser Abschnitt erläutert, wie das Tool Governor gestartet und gestoppt wird, und beschreibt die Aktivitäten des Governor-Dämons.

Starten und Stoppen von Governor

Das Dienstprogramm Governor überwacht Anwendungen, die eine Verbindung zu einer Datenbank herstellen, und ändert ihre Funktionsweise gemäß den Regeln, die Sie in einer Konfigurationsdatei von Governor für diese Datenbank angeben.

Voraussetzungen:

Bevor Sie Governor starten, müssen Sie die Konfigurationsdatei erstellen.

Einschränkungen:

Zum Starten oder Stoppen von Governor benötigen Sie die Berechtigung *sysadm* oder *sysctrl*.

Vorgehensweise:

Gehen Sie wie folgt vor, um Governor zu starten oder zu stoppen:

1. Führen Sie zum Starten von Governor den Befehl *db2gov* in der DB2-Befehlszeile aus. Geben Sie die folgenden erforderlichen Parameter an:
   - START *datenbankname*

     Der von Ihnen angegebene Datenbankname muss mit dem Namen der Datenbank in der Konfigurationsdatei übereinstimmen, die Sie angeben. Stimmen die Namen nicht überein, wird ein Fehler zurückgegeben. Beachten Sie, dass bei der Ausführung von Governor für mehrere Datenbanken für jede Datenbank Dämonen gestartet werden.
   - *konfig_dateiname*

     Der Name der Konfigurationsdatei von Governor für diese Datenbank. Wenn sich die Datei nicht an der Standardposition befindet, d. h. im Verzeichnis `sqllib`, müssen Sie außer dem Dateinamen auch den Pfad mit angeben.
   - *protokolldateiname*

     Der Basisname der Protokolldatei für diesen Governor. In einer partitionierten Datenbank wird die Partitionsnummer für jede Partition hinzugefügt, in der ein Dämon für dieses Exemplar von Governor ausgeführt wird.

   Zum Starten von Governor in einer einzelnen Partition für eine partitionierte Datenbank fügen Sie die Option *nodenum* hinzu. Um zum Beispiel Governor für eine Datenbank mit dem Namen *sales* nur auf Knoten 3 einer partitionierten Datenbank mit einer Konfigurationsdatei mit dem Namen *salescfg* und einer Protokolldatei mit dem Namen *saleslog* zu starten, geben Sie den folgenden Befehl ein:

   ```
   db2gov START sales nodenum 3 salescfg saleslog
   ```

Wenn Governor in allen Partitionen der Datenbank *sales* gestartet werden soll, geben Sie den folgenden Befehl ein:

```
db2gov START sales  salescfg saleslog
```

2. Zum Stoppen von Governor geben Sie den Befehl *db2gov* mit der Option STOP ein.

   Wenn Governor zum Beispiel in allen Partitionen der Datenbank *sales* gestoppt werden soll, geben Sie den folgenden Befehl ein:

   ```
   db2gov STOP sales
   ```

   Soll Governor nur in Partition 3 gestoppt werden, geben Sie den folgenden Befehl ein:

   ```
   db2gov START sales nodenum 3
   ```

Zugehörige Konzepte:

- „Das Dienstprogramm Governor" auf Seite 313
- „Der Governor-Dämon" auf Seite 315

Zugehörige Referenzen:

- „db2gov - DB2 Governor Command" in *Command Reference*

Der Governor-Dämon

Wenn der Governor-Dämon gestartet wird, d. h. wenn entweder Sie das Dienstprogramm `db2gov` ausführen oder Governor aktiv wird (wake-up), führt er die folgende wiederkehrende Abfolge von Operationen aus.

1. Er prüft, ob seine Governor-Konfigurationsdatei geändert bzw. noch nicht gelesen wurde. Trifft eine der beiden Bedingungen zu, liest der Dämon die Regeln in der Datei. Dadurch können Sie das Verhalten des Governor-Dämons ändern, während er aktiv ist.
2. Er fordert Momentaufnahmeninformationen zur Ressourcennutzungsstatistik für jede Anwendung und jeden Agenten an, die bzw. der in der Datenbank arbeitet.

   Anmerkung: Auf einigen Plattformen sind die CPU-Statistikdaten auf dem DB2® Monitor nicht verfügbar. In diesem Fall stehen die Abrechnungsregel (account) und die CPU-Begrenzung (cpu) nicht zur Verfügung.
3. Er überprüft die Statistik für jede einzelne Anwendung anhand der Regeln in der Governor-Konfigurationsdatei. Wenn eine Regel auf eine Anwendung zutrifft, führt Governor die angegebene Aktion aus.

   Anmerkung: Der Governor vergleicht aufgelaufene Informationen mit den in der Konfigurationsdatei definierten Werten. Dies bedeutet: Falls die Konfigurationsdatei mit neuen Werten aktualisiert wird, die eine Anwendung möglicherweise bereits überschritten hat, werden die Governor-Regeln, die für die betreffende Überschreitung gelten, im nächsten Governor-Intervall unverzüglich auf die Anwendung angewandt.
4. Er schreibt für jede ausgeführte Aktion einen Datensatz in die Governor-Protokolldatei.

Anmerkung: Governor kann nicht zur Anpassung der Agentenprioritäten verwendet werden, wenn der Konfigurationsparameter *agentpri* des

Datenbankmanagers einen anderen Wert als den Systemstandardwert enthält. (Dies gilt nicht für Windows® NT-Plattformen.)

Wenn Governor seine Operationen abgeschlossen hat, wird er für die in der Konfigurationsdatei angegebene Zeitdauer inaktiviert. Nach Ablauf dieses Intervalls wird Governor wieder aktiviert (wake-up) und führt dieselbe Abfolge von Operationen erneut aus.

Trifft Governor auf einen Fehler oder ein Stoppsignal, führt er vor der Beendigung eine Bereinigung durch. Bei der Bereinigung werden mit Hilfe einer Liste von Anwendungen, deren Prioritäten geändert wurden, die Prioritäten aller Anwendungsagenten zurückgesetzt. Anschließend werden die Prioritäten derjenigen Agenten zurückgesetzt, die nicht mehr für eine Anwendung arbeiten. Dadurch wird sichergestellt, dass keine Agenten nach Beendigung von Governor mit anderen als den Standardprioritäten aktiv bleiben. Falls ein Fehler auftritt, schreibt Governor eine Nachricht in das Benachrichtigungsprotokoll für die Verwaltung, um seine abnormale Beendigung anzuzeigen.

Anmerkung: Obwohl der Governor-Dämon keine Datenbankanwendung ist und daher auch keine Verbindung (Connect) zur Datenbank unterhält, hat er dennoch eine Exemplarverbindung (Attach). Da er Anforderungen für Momentaufnahmen absetzen kann, kann der Governor-Dämon erkennen, wenn der Datenbankmanager beendet wird.

Zugehörige Konzepte:

- „Das Dienstprogramm Governor" auf Seite 313

Zugehörige Tasks:

- „Starten und Stoppen von Governor" auf Seite 314

Konfiguration von Governor

Dieser Abschnitt erläutert die Konfiguration von Governor zur Überwachung und Steuerung von Datenbankaktivitäten.

Konfigurieren von Governor

Zum Konfigurieren von Governor erstellen Sie eine Konfigurationsdatei, in der die Datenbank, die von einem Governor-Exemplar überwacht wird, und die Verwaltung von Abfragen festgelegt werden.

Die Konfigurationsdatei besteht aus einer Menge von Regeln. Die ersten drei Regeln geben die zu überwachende Datenbank, das Intervall, in dem Protokollsätze zu schreiben sind, und das Intervall an, in dem der Governor-Dämon zur Überwachung aktiv wird. Die übrigen Regeln geben an, wie der Datenbankserver zu überwachen ist und welche Aktionen unter bestimmten Umständen auszuführen sind.

Vorgehensweise:

Gehen Sie wie folgt vor, um eine Governor-Konfigurationsdatei zu erstellen:

1. Erstellen Sie in einem Verzeichnis, das von allen Datenbankmanagerpartitionen aus zugänglich ist bzw. für sie angehängt ist, eine ASCII-Datei mit einem

beschreibenden Namen. Zum Beispiel könnte eine Konfigurationsdatei für ein Governor-Exemplar, das eine Datenbank **sales** überwacht, den Namen *govcfgsales* haben.

2. Öffnen Sie die Datei in einem beliebigen Texteditor und geben Sie Konfigurationsinformationen und Aktionsbedingungen ein.

   Schließen Sie jede Regel mit einem Semikolon (;) ab. Die folgenden Konfigurationsdaten werden empfohlen:

   - dbname: Der Name oder Aliasname der zu überwachenden Datenbank.
   - account: Die Zahl Minuten, nach denen das Governor-Exemplar Statistikdaten zur CPU-Nutzung in die zugehörige Protokolldatei schreibt. Diese Option ist unter Windows NT nicht verfügbar.
   - interval: Die Zahl Sekunden, nach denen der Governor-Dämon aktiv wird, um die Überwachungsfunktion auszuführen. Wenn Sie kein Intervall angeben, wird der Standardwert von 120 Sekunden verwendet.

   Zum Beispiel könnten die ersten drei Regeln in der Konfigurationsdatei wie folgt aussehen:

   ```
   { Einmal pro Sekunde aktiv werden, Datenbankname lautet sales,
    Abrechnungsdatensätze alle 30 Minuten }
   interval 1; dbname sales; account 30;
   ```

   Fügen Sie Regeln hinzu, welche die zu überwachenden Bedingungen und die Aktion angeben, die auszuführen ist, wenn die jeweilige Regel als wahr ausgewertet wird. Sie könnten zum Beispiel wie folgt eine Regel hinzufügen, welche die Zeit, die eine Arbeitseinheit (UOW - Unit of Work) ausgeführt werden kann, bevor die Anwendung zwangsweise von der Datenbank getrennt wird, auf eine Stunde begrenzt:

   ```
   setlimit uowtime 3600 action force;
   ```

3. Sichern Sie die Datei.

Zugehörige Konzepte:

- „Die Konfigurationsdatei von Governor" auf Seite 317
- „Elemente für Governor-Regeln" auf Seite 320
- „Beispiel für eine Governor-Konfigurationsdatei" auf Seite 325
- „Governor-Protokolldateien" auf Seite 326

Zugehörige Referenzen:

- „db2gov - DB2 Governor Command" in *Command Reference*

Die Konfigurationsdatei von Governor

Beim Starten von Governor geben Sie die Konfigurationsdatei an, die die Regeln zum Überprüfen der in einer Datenbank aktiven Anwendungen enthält. Governor wertet jede Regel aus und führt die angegebene Aktion aus, wenn die entsprechende Regel zutrifft.

Wenn sich Ihre Anforderungen für die Regeln ändern, editieren Sie die Konfigurationsdatei, ohne Governor zu stoppen. Jeder Governor-Dämon erkennt die geänderte Datei und liest sie erneut.

Die Konfigurationsdatei muss in einem Verzeichnis erstellt werden, das für alle Datenbankpartitionen angehängt ist, so dass alle Governor-Dämonen der einzelnen Partitionen dieselbe Konfigurationsdatei lesen können.

Die Konfigurationsdatei besteht aus drei erforderlichen Regeln, die die zu überwachende Datenbank, das Intervall, in dem Protokolldatensätze geschrieben werden und das Inaktivierungsintervall der Governor-Dämonen festlegen. Im Anschluss an diese Parameter enthält die Konfigurationsdatei eine Reihe optionaler Regeln zur Anwendungsüberwachung und zugehöriger Aktionen. Die folgenden Anmerkungen gelten für alle Regeln:

- Kommentare werden mit geschweiften Klammern { } begrenzt.
- Die meisten Einträge können in Großbuchstaben, Kleinbuchstaben oder gemischt angegeben werden. Die Ausnahme ist der Anwendungsname, der als Argument für die Regel `applname` angegeben wird, bei dem die Groß-/Kleinschreibung zu beachten ist.
- Jede Regel endet mit einem Semikolon (;).

Erforderliche Regeln

Die folgenden Regeln geben die zu überwachende Datenbank und das Intervall an, in dem der Dämon nach jedem Durchlauf der Aktivitäten wieder aktiv wird. Jede dieser Regeln wird nur einmal in der Datei angegeben.

dbname
Der Name oder Aliasname der zu überwachenden Datenbank.

account *nnn*
Abrechnungsdatensätze, die statistische Daten zur CPU-Auslastung für jede Verbindung enthalten, werden jeweils nach Ablauf des in Minuten angegebenen Intervalls geschrieben.

Anmerkung: Diese Option ist in der Windows NT-Umgebung nicht verfügbar.

Findet eine kurze Verbindungssitzung vollständig innerhalb eines Abrechnungsintervalls statt, so wird kein Protokollsatz geschrieben. Wenn Protokollsätze geschrieben werden, enthalten sie CPU-Statistiken, die Aufschluss über die CPU-Verwendung der Verbindung seit dem vorangegangenen Protokollsatz geben. Wird Governor gestoppt und erneut gestartet, wird die CPU-Verwendung möglicherweise in zwei Protokollsätzen aufgezeichnet. Diese können über die Anwendungs-IDs in den Protokollsätzen ermittelt werden.

interval
Das Intervall in Sekunden, nach dem der Dämon jeweils aktiv wird. Wenn Sie kein Intervall angeben, wird der Standardwert von 120 Sekunden verwendet.

Regeln, die Aktionen festlegen

Im Anschluss an die erforderlichen Regeln können Sie Regeln hinzufügen, die angeben, wie Anwendungen zu behandeln sind. Diese Regeln bestehen aus kleineren Komponenten, den so genannten Regelklauseln. Bei ihrer Verwendung müssen die Klauseln in einer bestimmten Reihenfolge in der Regelanweisung wie folgt angegeben werden:

1. **desc** (optional): Ein in Anführungszeichen eingeschlossener Kommentar über die Regel.
2. **time** (optional): Die Tageszeit, zu der die Regel ausgewertet wird.
3. **authid** (optional): Mindestens eine Berechtigungs-ID, unter der die Anwendung Anweisungen ausführt.

4. **applname** (optional): Der Name der ausführbaren Datei oder der Objektdatei, welche die Verbindung zur Datenbank herstellt. Dieser Name ist von der Groß-/Kleinschreibung abhängig. Der Anwendungsname muss in doppelte Anführungszeichen gesetzt werden, wenn er Leerzeichen enthält.
5. **setlimit**: Die Grenzwerte, die von Governor geprüft werden. Dies kann einer von mehreren Werten sein, wie zum Beispiel die CPU-Zeit, die Anzahl zurückgegebener Zeilen oder die inaktive Zeit.
6. **action** (optional): Die Aktion, die auszuführen ist, wenn ein Grenzwert erreicht wird. Wenn keine Aktion angegeben wird, verringert Governor die Priorität von Agenten, die für die Anwendung aktiv sind, um den Wert 10, wenn ein Grenzwert erreicht wird. Aktionen, die an der Datenbank ausgeführt werden können, sind zum Beispiel eine Verringerung der Agentenpriorität, eine erzwungene Trennung der Verbindung der Anwendung zur Datenbank oder eine Einstellung von Zeitplanoptionen für die Operationen der Anwendung.

Zur Definition einer Regel kombinieren Sie die Regelklauseln, wobei Sie jede Klausel in einer Regel nur einmal verwenden, und schließen jede Regel mit einem Semikolon ab, wie in den folgenden Beispielen zu sehen ist:

```
desc "Keine Arbeitseinheit darf länger als 1 Stunde dauern"
setlimit uowtime 3600 action force;

desc "Beschränken der Nutzung von db2 CLP durch neuen Benutzer novice"
authid novice
applname db2bp.exe
setlimit cpu 5 locks 100 rowssel 250;
```

Wenn mehrere Regeln auf eine Anwendung zutreffen, werden alle Regeln angewendet. Gewöhnlich wird die Aktion, die dem zuerst angetroffenen Grenzwert einer Regel zugeordnet ist, auch zuerst angewendet. Ausnahmen treten dann auf, wenn Sie den Wert -1 für eine Klausel in einer Regel angeben. In diesem Fall kann ein in der Klausel der nachfolgenden Regel angegebener Wert nur den zuvor in der *gleichen* Klausel angegebenen Wert überschreiben: die übrigen Klauseln in der vorhergehenden Regel bleiben weiterhin gültig. Eine Regel gibt in den Klauseln `rowssel 100000 uowtime 3600` zum Beispiel an, dass die Priorität einer Anwendung verringert werden muss, wenn ihre abgelaufene Zeit 1 Stunde überschreitet oder die Anwendung mehr als 100 000 Zeilen auswählt. Eine nachfolgende Regel enthält die Klausel `uowtime -1`, um anzugeben, dass die gleiche Anwendung über unbegrenzte Zeit verfügen kann. Wenn in diesem Fall die Anwendung länger als 1 Stunde ausgeführt wird, wird ihre Priorität nicht geändert. Das heißt, die Klausel `uowtime -1` überschreibt die Klausel `uowtime 3600`. Wenn die Anwendung jedoch über 100 000 Zeilen auswählt, wird ihre Priorität herabgesetzt, weil die Klausel `rowssel 100000` weiterhin gültig ist.

Reihenfolge der Regelanwendung

Der Governor verarbeitet die Regeln in der Konfigurationsdatei vom Anfang der Datei nach unten. Wenn jedoch die Klausel **setlimit** einer späteren Regel weniger restriktiv ist als eine vorherige Regel, wird die restriktivere Regel immer noch angewendet. Zum Beispiel wird `admin` in der folgenden Konfigurationsdatei trotz der späteren Regel auf 5000 Zeilen begrenzt, da die erste Regel restriktiver ist.

```
desc "Erzwingen, dass alle 5000 oder mehr Zeilen auswählen"
setlimit rowssel 5000 action force;

desc "Benutzer admin das Auswählen von weiteren Zeilen erlauben"
authid admin
setlimit rowssel 10000 action force;
```

Um sicherzustellen, dass eine weniger restriktive Regel eine restriktivere Regel außer Kraft zu setzen, die in der Datei weiter oben steht, können Sie die Option -1 angeben, um die vorherige Regel vor dem Anwenden der neuen Regel zu löschen. Beispielsweise begrenzt die Anfangsregel in der folgenden Konfigurationsdatei alle Benutzer auf 5000 Zeilen. Die zweite Regel löscht diese Begrenzung für `admin` und die dritte Regel setzt die Begrenzung für `admin` auf 10000 Zeilen zurück.

```
desc "Erzwingen, dass alle 5000 oder mehr Zeilen auswählen"
setlimit rowssel 5000 action force;

desc "Begrenzung rowssel für admin löschen"
authid admin
setlimit rowssel -1;

desc "Nun die höhere Begrenzung rowssel für admin festlegen"
authid admin
setlimit rowssel 10000 action force;
```

Zugehörige Konzepte:

- „Elemente für Governor-Regeln" auf Seite 320
- „Beispiel für eine Governor-Konfigurationsdatei" auf Seite 325

Elemente für Governor-Regeln

Jede Regel in der Konfigurationsdatei von Governor wird aus Klauseln zusammengesetzt, welche die Bedingungen zur Anwendung der Regel und die Aktion angeben, die folgt, wenn die Regel als wahr ausgewertet wird. Die Klauseln müssen in der angezeigten Reihenfolge angegeben werden. In den Beschreibungen der Klauseln weisen geschweifte Klammern [ ] auf eine optionale Klausel hin.

Optionale Startelemente

[desc] Gibt eine Textbeschreibung der Regel an. Die Beschreibung muss in einfachen oder doppelten Anführungszeichen stehen.

[time] Gibt die Zeitspanne an, während der die Regel auswertet werden soll.

Die Zeitspanne muss im Format `time hh:mm hh:mm` angegeben werden, z. B. `time 8:00 18:00`. Wird diese Klausel nicht angegeben, ist die Regel 24 Stunden am Tag gültig.

[authid]

Gibt eine oder mehrere Berechtigungs-IDs (authid) an, mit denen die Anwendung ausgeführt wird. Mehrere Berechtigungs-IDs müssen durch ein Komma (,) voneinander getrennt werden, z. B. `authid gene, michael, james`. Wird diese Klausel in einer Regel nicht angegeben, gilt die Regel für alle Berechtigungs-IDs.

[applname]

Gibt den Namen der ausführbaren Datei (oder Objektdatei) an, welche die Verbindung zur Datenbank herstellt.

Mehrere Anwendungsnamen müssen durch ein Komma (,) voneinander getrennt werden, z. B. `applname db2bp, batch, geneprog`. Wird diese Klausel in einer Regel nicht angegeben, gilt die Regel für alle Anwendungsnamen.

Anmerkungen:

1. Bei Anwendungsnamen muss Groß-/Kleinschreibung beachtet werden.

2. Der Datenbankmanager schneidet alle Anwendungsnamen auf 20 Zeichen ab. Stellen Sie sicher, dass die zu prüfende Anwendung mit den ersten 20 Zeichen ihres Anwendungsnamens eindeutig identifiziert wird. Ist das nicht der Fall, wird möglicherweise unbeabsichtigt eine andere Anwendung überprüft.

   Anwendungsnamen in der Governor-Konfigurationsdatei werden auf 20 Zeichen abgeschnitten, damit sie mit ihrer internen Darstellung übereinstimmen.

Grenzwertklauseln

setlimit

Gibt mindestens einen Grenzwerte an, den Governor überprüfen soll. Die Grenzwerte dürfen nur den Wert -1 oder Werte größer als 0 annehmen (z. B. `cpu -1 locks 1000 rowssel 10000`). Sie müssen mindestens eine der Begrenzungen (cpu, locks, rowsread, uowtime) angeben. Jede nicht angegebene Begrenzung wird durch die jeweilige Regel nicht eingeschränkt. Governor kann folgende Begrenzungen überprüfen:

cpu *nnn*

Gibt die Anzahl der CPU-Sekunden an, die eine Anwendung nutzen kann. Wenn Sie den Wert `-1` angeben, schränkt Governor die CPU-Nutzung der Anwendung nicht ein.

Anmerkung: Diese Option ist in der Windows® NT;-Umgebung nicht verfügbar.

locks *nnn*

Gibt die Anzahl der Sperren an, die eine Anwendung halten kann. Wenn Sie den Wert `-1` angeben, schränkt Governor die Anzahl der durch die Anwendung gehaltenen Sperren nicht ein.

rowssel *nnn*

Gibt die Anzahl von Zeilen an, die an die Anwendung zurückgegeben werden. Dieser Wert ist nur auf dem Koordinatorknoten ungleich Null. Wenn Sie den Wert `-1` angeben, schränkt Governor die Anzahl der auswählbaren Zeilen nicht ein.

uowtime *nnn*

Gibt die Zeitspanne in Sekunden an, die verstreichen kann, nachdem eine Arbeitseinheit (UOW) zum ersten Mal aktiv wird. Wenn Sie den Wert `-1` angeben, wird die abgelaufene Zeit nicht eingeschränkt.

Anmerkung: Wenn Sie die API `sqlmon` (den Datenbanksystemmonitor-Schalter) zum Inaktivieren des Schalters der Arbeitseinheit verwendet haben, wird die Fähigkeit von Governor beeinträchtigt, Anwendungen auf der Grundlage der abgelaufenen Zeit der Arbeitseinheit zu überprüfen. Governor verwendet das Überwachungsprogramm zum Sammeln von Systeminformationen. Wenn Sie die Schalter in der Konfigurationsdatei des Datenbankmanagers ausschalten, wird dieses Programm für das gesamte Exemplar abgeschaltet. Dadurch erhält Governor diese Informationen nicht mehr.

idle *nnn*

Gibt die für eine Verbindung zulässige Leerlaufzeit in Sekunden

an, bevor eine Aktion ausgeführt wird. Wenn Sie den Wert -1 angeben, wird die Leerlaufzeit der Verbindung nicht eingeschränkt.

rowsread *nnn*
Gibt die Anzahl der von einer Anwendung auswählbaren Zeilen an. Wenn Sie den Wert -1 angeben, kann die Anwendung eine unbegrenzte Anzahl an Zeilen auswählen.

Anmerkung: Diese Begrenzung ist nicht mit rowssel identisch. Der Unterschied besteht darin, dass sich rowsread auf die Anzahl der Zeilen bezieht, die gelesen werden mussten, um eine Ergebnismenge zurückgeben zu können. Die Anzahl der gelesenen Zeilen schließt Lesevorgänge der Katalogtabellen durch die Steuerkomponente ein und kann durch die Verwendung von Indizes verringert werden.

Aktionsklauseln

[action]
Gibt die Aktion an, die ausgeführt werden muss, wenn ein angegebener Grenzwert überschritten wird. Sie können folgende Aktionen angeben:

Anmerkung: Wenn ein Grenzwert überschritten wird und die Aktionsklausel nicht angegeben ist, verringert Governor die Priorität der für die Anwendung aktiven Agenten um 10.

nice *nnn*
Gibt eine Änderung der Priorität der für die Anwendung aktiven Agenten an. Gültige Werte: –20 bis +20.

Dieser Parameter ist unter folgenden Voraussetzungen wirksam:
- Auf UNIX®-Plattformen muss der Parameter *agentpri* des Datenbankmanagers den Standardwert aufweisen. Andernfalls überschreibt er die Prioritätsklausel.
- Auf Windows-Plattformen kann der Datenbankmanagerparameter *agentpri* gemeinsam mit der Aktion *priority* verwendet werden.

force Gibt an, dass der Agent, der die Anwendung bedient, zum Trennen der Verbindung gezwungen wird. (Setzt die Anweisung FORCE APPLICATION ab, um den koordinierenden Agenten zu beenden.)

schedule [class]
Durch eine Zeitzuweisung werden die Prioritäten der für die Anwendungen aktiven Agent verbessert, um die durchschnittlichen Antwortzeiten zu minimieren, ohne Anwendungen zu benachteiligen.

Governor wählt die Anwendungen mit den jeweils höchsten Werten für die Zeitzuweisung nach den folgenden drei Kriterien aus:
- Die Anwendung, welche die meisten Sperren hält

  Diese Auswahl erfolgt mit dem Ziel, die Anzahl von Situationen zu verringern, in denen auf Sperren gewartet werden muss (lockwaits).
- Die älteste Anwendung
- Die Anwendung mit der kürzesten geschätzten verbleibenden Ausführungszeit

Diese Auswahl ist ein Versuch, möglichst viele kurzzeitig aktive Anweisungen während des Intervalls zum Abschluss zu bringen.

Die drei bei den jeweiligen Kriterien höchstplatzierten Anwendungen erhalten höhere Prioritäten als alle anderen Anwendungen. Das bedeutet, der höchstplatzierten Anwendung in jeder Kriteriumsgruppe wird die höchste Priorität, den nächsthöchsten Anwendungen die zweithöchste Priorität und den dritthöchsten Anwendungen die dritthöchste Priorität erteilt. Wenn eine einzelne Anwendung auf einem der drei höchsten Plätze in mehreren Kategorien rangiert, erhält sie die entsprechende Priorität für das Kriterium, bei dem sie den höchsten Rang innehat. Die nächsthöchste Anwendung erhält die nächsthöchste Priorität für das andere Kriterium. Wenn zum Beispiel Anwendung A die meisten Sperren hält, jedoch die drittkürzeste geschätzte verbleibende Ausführungszeit hat, erhält sie die höchste Priorität für das erste Kriterium. Die Anwendung, die mit der kürzesten geschätzten Ausführungszeit auf Platz vier rangiert erhält dadurch die dritthöchste Priorität für dieses zweite Kriterium.

Die Anwendungen, die von dieser Governor-Regel ausgewählt werden, werden in bis zu drei Klassen unterteilt. Für jede Klasse wählt Governor neun Anwendungen aus, welche nach den oben aufgeführten Kriterien jeweils die drei höchstplatzierten Anwendungen jeder Klasse sind. Wenn Sie die Klassenoption (class) angeben, werden alle Anwendungen, die von dieser Regel ausgewählt werden, als eine einzige Klasse betrachtet, wobei neun Anwendungen ausgewählt werden und wie oben beschrieben höhere Prioritäten erhalten.

Wenn eine Anwendung in mehreren Governor-Regeln ausgewählt wird, wird sie von der letzten Regel regiert, in der sie ausgewählt wird.

Anmerkung: Wenn Sie die API `sqlmon` (den Datenbanksystemmonitor-Schalter) zum Inaktivieren des Schalters der Anweisung verwendet haben, wird die Fähigkeit von Governor beeinträchtigt, Anwendungen auf der Grundlage der abgelaufenen Zeit der Anweisung zu überprüfen. Governor verwendet das Überwachungsprogramm zum Sammeln von Systeminformationen. Wenn Sie die Schalter in der Konfigurationsdatei des Datenbankmanagers ausschalten, wird dieses Programm für das gesamte Exemplar abgeschaltet. Dadurch erhält Governor diese Informationen nicht mehr.

Eine Zeitzuweisungsaktion kann folgenden Zwecken dienen:

- Anwendungen in unterschiedlichen Gruppen erhalten Zeit zugewiesen, ohne dass diese Zeit gleichmäßig unter allen Anwendungen aufgeteilt wird.

  Wenn beispielsweise 14 Anwendungen (3 kurze, 5 mittlere und 6 lange) gleichzeitig aktiv sind, haben möglicherweise alle eine schlechte Antwortzeit, weil sie die CPU-Zeit teilen. Der Datenbankadministrator kann zwei Gruppen einrichten, eine mit mittleren Anwendungen und eine mit langen Anwendungen. Mit

Hilfe der Prioritäten kann Governor alle kurzen Anwendungen ausführen und sicherstellen, dass höchstens drei mittlere und drei lange Anwendungen gleichzeitig aktiv sind. Zu diesem Zweck enthält die Governor-Konfigurationsdatei eine Regel für mittlere und eine Regel für lange Anwendungen.

Das folgende Beispiel zeigt einen Abschnitt einer Governor-Konfigurationsdatei, der dies veranschaulicht:

```
desc "Mittellange Anwendungen in 1 Zeitzuweisungsklasse
zusammenfassen"
applname medq1, medq2, medq3, medq4, medq5
setlimit cpu -1
action schedule class;

desc "Lange Anwendungen in 1 Zeitzuweisungsklasse zusammenfassen"
applname longq1, longq2, longq3, longq4, longq5, longq6
setlimit cpu -1
action schedule class;
```

- Unterschiedliche Benutzergruppen (z. B. Abteilungen in Unternehmen) erhalten gleiche Kriterien für die Vergabe von Prioritäten.

  Wenn eine Gruppe eine Vielzahl von Anwendungen ausführt, kann der Administrator sicherstellen, dass andere Gruppen dennoch akzeptable Antwortzeiten für ihre Anwendungen erhalten. Sind zum Beispiel drei Abteilungen beteiligt (Finanzen, Lagerbestand und Planung), kann man alle Benutzer der Finanzabteilung in eine Gruppe legen, alle Benutzer aus dem Lager in eine andere und alle Planer in die dritte. Dann werden die Verarbeitungskapazitäten in etwa gleichmäßig unter den drei Abteilungen aufgeteilt. Das folgende Beispiel zeigt einen Abschnitt einer Governor-Konfigurationsdatei, der dies veranschaulicht:

  ```
  desc "Benutzer der Finanzwesenabteilung in Klasse zusammenfassen"
  authid tom, dick, harry, mo, larry, curly
  setlimit cpu -1
  action schedule class;

  desc "Benutzer der Lagerabteilung in Klasse zusammenfassen"
  authid pat, chris, jack, jill
  setlimit cpu -1
  action schedule class;

  desc "Benutzer der Planungsabteilung in Klasse zusammenfassen"
  authid tara, dianne, henrietta, maureen, linda, candy
  setlimit cpu -1
  action schedule class;
  ```

- Governor bindet alle Anwendungen in eine Zeitzuweisung ein.

  Wenn die Option *class* nicht mit der Aktion angegeben wird, erstellt Governor eigene Klassen nach der Anzahl der aktiven Anwendungen, für die diese Aktion gilt, und ordnet die Anwendungen in verschiedene Klassen ein. Dies erfolgt entsprechend der Aufwandsschätzung durch den DB2-Abfragecompiler für die Abfrage, die von der Anwendung ausgeführt wird. Der Administrator kann alle Anwendungen für die Zeitzuweisung (Scheduling) auswählen, indem er nicht angibt, welche Anwendungen auszuwählen sind. Die Klauseln *applname* und *authid* werden also nicht angegeben, und die Klausel *setlimit* verursacht keine Einschränkungen.

Anmerkung: Wenn ein Grenzwert überschritten wird und die Aktionsklausel nicht angegeben ist, verringert Governor die Priorität der für die Anwendung aktiven Agenten.

Zugehörige Konzepte:

- „Die Konfigurationsdatei von Governor" auf Seite 317
- „Beispiel für eine Governor-Konfigurationsdatei" auf Seite 325

Zugehörige Tasks:

- „Konfigurieren von Governor" auf Seite 316

Beispiel für eine Governor-Konfigurationsdatei

Das folgende Beispiel zeigt eine Governor-Konfigurationsdatei, die verschiedene Regeln mit Aktionen definiert:

```
{ Einmal pro Sekunde aktiv werden, Datenbankname lautet ibmsampl,
 Abrechnungsdatensätze alle 30 Minuten }
interval 1; dbname ibmsampl; account 30;

desc "CPU-Einschränkungen gelten 24 Stunden pro Tag für alle"
setlimit cpu 600 rowssel 1000000 rowsread 5000000;

desc "Keine Arbeitseinheit darf länger als 1 Stunde dauern"
setlimit uowtime 3600 action force;

desc 'Verlangsamen einer Untermenge von Anwendungen'
applname jointA, jointB, jointC, quryA
setlimit cpu 3 locks 1000 rowssel 500 rowsread 5000;

desc "Governor soll diese 6 langen Anwendungen in 1 Prioritätenklasse einordnen"
applname longq1, longq2, longq3, longq4, longq5, longq6
setlimit cpu -1
action schedule class;

desc "Zeitzuweisung für alle Anwendungen von der Planungsabteilung"
authid planid1, planid2, planid3, planid4, planid5
setlimit cpu -1
action schedule;

desc "Einordnen aller CPU-Fresser in eine Klasse zur Verbrauchssteuerung"
setlimit cpu 3600
action schedule class;

desc "Beschränken der Nutzung von db2 CLP durch neuen Benutzer novice"
authid novice
applname db2bp.exe
setlimit cpu 5 locks 100 rowssel 250;

desc "Zur Tagesarbeitszeit darf keine Anwendung länger als 10 Sek. aktiv sein"
time 8:30 17:00 setlimit cpu 10 action force;

desc "Einige Benutzer, die mit Leistungsoptimierung befasst sind, dürfen
      einige Ihrer Anwendungen in der Mittagspause durchführen"
time 12:00 13:00 authid ming, geoffrey, john, bill
applname tpcc1, tpcc2, tpcA, tpvG setlimit cpu 600 rowssel 120000 action force;

desc "Einige Benutzer sollten nicht eingeschränkt werden -- Datenbankadministrator
      und einige andere. Da dies die letzte Angabe in der Datei ist,
      wird zuvor Angegebenes hierdurch überschrieben."
authid gene, hershel, janet setlimit cpu -1 locks -1 rowssel -1 uowtime -1;
```

```
desc "Erhöhen der Priorität einer wichtigen Anwendung, so dass sie immer
      schnell verarbeitet wird"
applname V1app setlimit cpu 1 locks 1 rowssel 1 action priority -20;
```

Zugehörige Konzepte:

- „Die Konfigurationsdatei von Governor" auf Seite 317
- „Elemente für Governor-Regeln" auf Seite 320

Zugehörige Tasks:

- „Konfigurieren von Governor" auf Seite 316

Verwendung der Governor-Protokolldateien

Dieser Abschnitt beschreibt die Governor-Protokolldateien und erläutert, wie diese Dateien zum Abrufen von Informationen abgefragt werden können.

Governor-Protokolldateien

Wenn ein Governor-Dämon eine Aktion ausführt, schreibt er einen Datensatz in seine Protokolldatei. Zu diesen Aktionen gehören:

- Erzwungene Beendigung einer Anwendung
- Lesen der Governor-Konfigurationsdatei
- Ändern einer Anwendungspriorität
- Feststellen einer Fehler- oder Warnbedingung
- Starten oder Stoppen

Jeder Governor-Dämon besitzt eine eigene Protokolldatei. Getrennte Protokolldateien vermeiden Engpässe durch Sperrungen von Dateien, die auftreten könnten, wenn viele Governor-Dämonen gleichzeitig in dieselbe Datei schreiben würden. Mit dem Dienstprogramm `db2govlg` können Sie die Protokolldateien zusammenfügen und Abfragen ausführen.

Die Protokolldateien werden im Unterverzeichnis `log` des Verzeichnisses `sqllib` gespeichert. Für Windows® NT gilt jedoch die Ausnahme, dass sich das Unterverzeichnis `log` unterhalb des Exemplarverzeichnisses befindet. Wenn Sie Governor mit dem Befehl `db2gov` starten, geben Sie den Basisnamen für die Protokolldatei an. Stellen Sie sicher, dass der Protokolldateiname den Datenbanknamen enthält, um die Protokolldateien in jeder Partition unterscheiden zu können, in der Governor ausgeführt wird. Um in einer Umgebung mit partitionierten Datenbanken sicherzustellen, dass der Dateiname eindeutig ist, wird die Partitionsnummer der Partition, in der der Governor-Dämon ausgeführt wird, automatisch an den Protokolldateinamen angefügt.

Datensatzformat der Protokolldatei

Ein Datensatz in der Protokolldatei hat folgendes Format:

```
Datum Uhrzeit Knoten-Nr. Satztyp Nachricht
```

Anmerkung: Das Format der Felder *Datum* und *Uhrzeit* ist jjjj-mm-tt hh.mm.ss. Sie können die Protokolldateien in jeder Datenbankpartition durch Sortieren nach diesem Feld zusammenfügen.

Das Feld *Knoten-Nr.* gibt die Nummer der Datenbankpartition an, in der Governor aktiv ist.

Das Feld *Satztyp* enthält unterschiedliche Werte, je nach Typ des in das Protokoll geschriebenen Eintrags. Folgende Werte können eingetragen werden:
- `START`: Governor wurde gestartet.
- `STOP`: Governor wurde gestoppt.
- `FORCE`: Eine Anwendung wurde zwangsweise getrennt.
- `NICE`: Die Priorität einer Anwendung wurde geändert.
- `ERROR`: Ein Fehler ist aufgetreten.
- `WARNING`: Eine Warnung ist aufgetreten.
- `READCFG`: Governor hat die Konfigurationsdatei gelesen.
- `ACCOUNT`: Die Abrechnungsstatistik der Anwendung.
- `SCHEDGRP`: Eine Änderung der Agentenpriorität ist aufgetreten.

Einige dieser Werte werden weiter unten näher beschrieben.

START
Der Datensatz START wird geschrieben, wenn der Governor gestartet wird. Er hat folgendes Format:

```
Datenbank = <datenbankname>
```

STOP Der Datensatz STOP wird geschrieben, wenn der Governor gestoppt wird. Er hat folgendes Format:

```
Datenbank = <datenbankname>
```

FORCE
Der Datensatz wird immer dann ausgeschrieben, wenn der Governor feststellt, dass eine Anwendung in der Konfigurationsdatei des Governor erzwungen werden soll, wie dies für eine Regel erforderlich ist. Der Datensatz FORCE hat folgendes Format:

```
<anwendungsname> <authentifizierungs_id> <anwendungs_id>
<koordinierende_partition> <konfigurationszeile>
<einschränkung_überschritten>
```

Dabei gilt:

<koordinierende_partition>
Gibt die Zahl der koordinierenden Partition der Anwendung an.

<konfigurationszeile>
Gibt die Zeilennummer in der Governor-Konfigurationsdatei an, in der sich die Regel befindet, die das Erzwingen der Anwendung verursacht.

<einschränkung_überschritten>
Stellt Einzelangaben zur Verfügung, wie die Regel überschritten wurde. Dies können die folgenden Werte sein:
- `CPU`: Die gesamte USR CPU plus die SYS CPU-Zeit in Sekunden.
- `Locks`: Die Summe der Sperren, die von der Anwendung gehalten werden
- `Rowssel`: Die Summe der Zeilen, die von der Anwendung ausgewählt werden
- `Rowsread`: Die Summe der Zeilen, die von der Anwendung gelesen werden
- `Idle`: Zeit, in der die Anwendung inaktiv war
- `ET`: Zeit, die seit dem Start der aktuellen Arbeitseinheit der Anwendung vergangen ist (das uowtime-Zeitlimit wurde überschritten)

NICE Der Datensatz NICE wird geschrieben, wenn die Priorität einer Anwendung über eine Prioritätsaktion geändert wird, die in der Konfigurationsdatei des Governor angegeben wird. Der Datensatz NICE hat folgendes Format:

```
<anwendunsname> <authentifizierungs_id> <anwendungs_id> <nice_wert>
(<konfigurationszeile>) <einschränkung_überschritten>
```

Dabei gilt Folgendes:

<nice_wert>
Gibt das Inkrement (oder Dekrement) an, das für den Agentenprozess der Anwendung zum Prioritätswert gemacht wird.

<konfigurationszeile>
Gibt die Zeilennummer in der Konfigurationsdatei des Governor an, in der sich die Regel befindet, die das Ändern der Anwendungspriorität verursacht.

<einschränkung_überschritten>
Stellt Einzelangaben zur Verfügung, wie die Regel überschritten wurde. Dies können die folgenden Werte sein:

- `CPU`: Die gesamte USR CPU plus die SYS CPU-Zeit in Sekunden.
- `Locks`: Die Summe der Sperren, die von der Anwendung gehalten werden
- `Rowssel`: Die Summe der Zeilen, die von der Anwendung ausgewählt werden
- `Rowsread`: Die Summe der Zeilen, die von der Anwendung gelesen werden
- `Idle`: Zeit, in der die Anwendung inaktiv war
- `ET`: Zeit, die seit dem Start der aktuellen Arbeitseinheit der Anwendung vergangen ist (das uowtime-Zeitlimit wurde überschritten)

ERROR
Der Datensatz ERROR wird geschrieben, wenn der Governor-Dämon abgeschaltet werden muss.

WARNING
Der Datensatz WARNING wird in den folgenden Situationen in das Governor-Protokoll geschrieben:

- Die API sqlefrce wurde aufgerufen, um eine Anwendung zu erzwingen aber es wurde ein positiver SQLCODE-Wert zurückgegeben.
- Ein Momentaufnahmenaufruf gab einen positiven SQLCODE-Wert zurück, der nicht 1611 ("SQL1661 Es wurden keine Daten zurückgegeben") war.
- Ein Momentaufnahmenaufruf gab einen negativen SQLCODE-Wert zurück, der nicht –1224 ("SQL1224N Ein Datenbankagent konnte nicht für die Anforderung gestartet werden, oder er wurde aufgrund eines Systemabschlusses der Datenbank bzw. durch den Befehl FORCE beendet") oder –1032 ("SQL1032N Der Befehl DB2START wurde nicht abgesetzt") war. Diese Rückkehrcodes treten auf, wenn ein zuvor aktives Exemplar inaktiv wurde.
- In einer UNIX®-Umgebung wurde ein Versuch unternommen, eine Signalroutine zu installieren. Der Versuch ist fehlgeschlagen.

ACCOUNT

Der Datensatz ACCOUNT wird in den folgenden Situationen in das Governor-Protokoll geschrieben:

- Der Wert von agent_usr_cpu oder agent_sys_cpu für diese Anwendung wurde geändert seit der letzte Datensatz ACCOUNT für diese Anwendung geschrieben wurde.
- Es wird festgestellt, dass die Anwendung nicht mehr aktive ist.

Der Datensatz ACCOUNT hat folgendes Format:

<authentifizierungs_id> <anwendungs_id> <anwendungsname> <verbindungsdauer> <agent_usr_cpu delta> <agent_sys_cpu delta>

SCHEDGRP

Der Datensatz SCHEDGRP wird immer unter den folgenden Umständen geschrieben:

- Die Anwendung wird einer Zeitplanungsgruppe hinzugefügt.
- Die Anwendung wird von einer Zeitplanungsgruppe in eine andere versetzt.

Der Datensatz SCHEDGRP hat folgendes Format:

```
<anwendungsname> <authentifizierungs_id> <anwendungs_id>
<konfigurationszeile> <einschränkung_überschritten>
```

Dabei gilt Folgendes:

<konfigurationszeile>
Gibt die Zeilennummer in der Governor-Konfigurationsdatei an, in der sich die Regel befindet, die das Terminieren der Anwendung verursacht.

<einschränkung_überschritten>
Stellt Einzelangaben zur Verfügung, wie die Regel überschritten wurde. Dies können die folgenden Werte sein:

- `CPU`: Die gesamte USR CPU plus die SYS CPU-Zeit in Sekunden.
- `Locks`: Die Summe der Sperren, die von der Anwendung gehalten werden
- `Rowssel`: Die Summe der Zeilen, die von der Anwendung ausgewählt werden
- `Rowsread`: Die Summe der Zeilen, die von der Anwendung gelesen werden
- `Idle`: Zeit, in der die Anwendung inaktiv war
- `ET`: Zeit, die seit dem Start der aktuellen Arbeitseinheit der Anwendung vergangen ist (das uowtime-Zeitlimit wurde überschritten)

Da standardisierte Werte geschrieben werden, können Sie die Protokolldateien nach unterschiedlichen Aktionstypen abfragen. Das Feld *Nachricht* enthält andere nicht standardisierte Informationen, die sich nach dem im Feld *Satztyp* angegebenen Wert ändern.Ein Eintrag des Typs `FORCE` oder `NICE` beispielsweise zeigt Informationen zur Anwendung im Feld *Nachricht* an, während ein Eintrag des Typs `ERROR` eine Fehlernachricht enthält.

Eine Beispielprotokolldatei könnte folgendermaßen aussehen:

```
1995-12-11 14.54.52    0 START      Database = TQTEST
1995-12-11 14.54.52    0 READCFG    Config = /u/db2instance/sqllib/tqtest.cfg
1995-12-11 14.54.53    0 ERROR      SQLMON Error: SQLCode = -1032
1995-12-11 14.54.54    0 ERROR      SQLMONSZ Error: SQLCode = -1032
```

Zugehörige Konzepte:

- „Das Dienstprogramm Governor" auf Seite 313
- „Abfragen von Governor-Protokolldateien" auf Seite 330

Abfragen von Governor-Protokolldateien

Jeder Governor-Dämon schreibt in eine eigene Protokolldatei. Mit dem Dienstprogramm `db2govlg` können Sie die Protokolldatei abfragen. Sie können die Protokolldateien einer einzelnen oder aller Datenbankpartitionen auflisten, sortiert nach Datum und Uhrzeit. Außerdem können Sie Abfragen auf der Grundlage des Protokollfelds *RecType* (Satztyp) durchführen. `db2govlg` hat folgende Syntax:

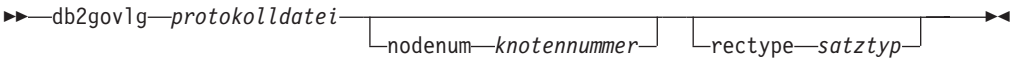


Abbildung 25. Syntax von db2govlg

Der Befehl hat die folgenden Parameter:

protokolldatei
Der Basisdateiname der Protokolldatei (oder Dateien), die Sie abfragen möchten

nodenum **knotennummer**
Die Knotennummer der Datenbankpartition, auf der Governor aktiv ist

rectype **satztyp**
Der Satztyp, die Sie abfragen möchten. Es gibt folgende Satztypen:
- START
- READCFG
- STOP
- FORCE
- NICE
- ERROR
- WARNING
- ACCOUNT

Für die Verwendung dieses Dienstprogramms gibt es keinerlei Berechtigungseinschränkungen. Dadurch können alle Benutzer abfragen, ob Governor ihre Anwendung geändert hat. Wenn Sie den Zugriff auf dieses Dienstprogramm beschränken möchten, können Sie die Gruppenberechtigungen für die Datei `db2govlg` ändern.

Zugehörige Konzepte:

- „Das Dienstprogramm Governor" auf Seite 313
- „Governor-Protokolldateien" auf Seite 326

Kapitel 10. Skalieren der Konfiguration

Dieses Kapitel beschreibt, wie Sie die Datenbankkapazität verwalten können, indem Sie in erster Linie Datenbankpartitionen hinzufügen und löschen. Weitere Methoden zur Erhöhung der Kapazität sind zum Beispiel das Hinzufügen von CPUs und Speicher.

Verwaltung der Datenbankserverkapazität

Wenn die Kapazität des Datenbankmanagers nicht Ihren gegenwärtigen oder zukünftigen Anforderungen genügt, können Sie die Kapazität des Datenbankmanagers wie folgt erweitern:

- Hinzufügen von Plattenspeicherplatz und Erstellen zusätzlicher Behälter
- Hinzufügen von Hauptspeicher

Wenn diese einfachen Strategien nicht die erforderliche Kapazität erbringen, können Sie folgende Methoden in Betracht ziehen:

- Hinzufügen von Prozessoren

  Wenn eine Konfiguration mit Einzelpartition und einem Einzelprozessor bis zur maximalen Kapazitätsgrenze ausgelastet wird, können Sie entweder Prozessoren oder Partitionen hinzufügen. Der Vorteil hinzugefügter Prozessoren liegt in der größeren Verarbeitungskapazität. In einem SMP-System benutzen Prozessoren Hauptspeicher und Speichersystemressourcen gemeinsam. Alle Prozessoren befinden sich in einem System, so dass keine Aufwände zu beachten sind, wie zum Beispiel die Kommunikation oder die Koordination von Tasks zwischen Systemen. Die Dienstprogramme von DB2®, beispielsweise zum Laden, Sichern und Wiederherstellen, können die zusätzlichen Prozessoren vorteilhaft nutzen. DB2 Universal Database™ unterstützt eine solche Umgebung.

  Anmerkung: Einige Betriebssysteme, wie zum Beispiel Solaris Operating Environment, können Prozessoren dynamisch online und offline schalten.

  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:

  – Grad der Parallelität (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 Registriervariablen DB2TCPCONNMGRS.

- Hinzufügen physischer Knoten

  Wenn Ihr Datenbankmanager zurzeit partitioniert ist, können Sie sowohl die Datenspeicherkapazitäten als auch die Verarbeitungskapazitäten erhöhen, indem Sie separate physischen Knoten aus Einzelprozessor- oder Multiprozessormaschinen hinzufügen. Der Hauptspeicher und die Speichersystemressourcen

jedes Knotens werden nicht mit den anderen Knoten gemeinsam benutzt. Obwohl das Hinzufügen von Knoten mit Problemen hinsichtlich der Kommunikation und der Taskkoordination verbunden sein kann, bietet diese Wahl den Vorteil, Daten und Benutzerzugriffe auf mehr als ein System verteilen zu können. DB2 Universal Database unterstützt eine solche Umgebung.

Sie können Knoten hinzufügen, während das Datenbankmanagersystem aktiv ist oder während es gestoppt ist. Wenn Sie Knoten hinzufügen, während das System aktiv ist, müssen Sie das System jedoch stoppen und erneut starten, bevor Datenbanken auf den neuen Knoten migriert werden.

Wenn Sie Ihr System durch 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 sowie das Sichern oder Wiederherstellen der Datenbank, im Klaren sein.

Wenn Sie eine neue Datenbankpartition hinzufügen, können Sie eine Datenbank, die die neue Partition nutzt, erst dann löschen oder erstellen, wenn die Aktion abgeschlossen und der neue Server erfolgreich in das System integriert ist.

Zugehörige Konzepte:

• „Partitionen in einer partitionierten Datenbank" auf Seite 332

Partitionen in einer partitionierten Datenbank

Sie können einem System mit einer partitionierten Datenbank Datenbankpartitionen hinzufügen, wenn es aktiv oder gestoppt ist. Da das Hinzufügen eines neuen Servers sehr zeitaufwendig sein kann, ist es vielleicht wünschenswert, dies zu tun, wenn der Datenbankmanager bereits aktiv ist.

Verwenden Sie den Befehl ADD DBPARTITIONNUM, um einem System eine Datenbankpartition hinzuzufügen. Dieser Befehl kann folgendermaßen aufgerufen werden:

- Als Option im Befehl `db2start`
- Mit dem Befehl ADD DBPARTITIONNUM über den Befehlszeilenprozessor
- Mit der API-Funktion `sqleaddn`
- Mit der API-Funktion `sqlepstart`

Wenn Ihr System gestoppt ist, verwenden Sie den Befehl `db2start`. Wenn es aktiv ist, können Sie eine der anderen Möglichkeiten verwenden.

Wenn Sie dem System mit dem Befehl ADD DBPARTITIONNUM eine neue Datenbankpartition hinzufügen, werden alle in diesem Exemplar vorhandenen Datenbanken auf die neue Datenbankpartition erweitert. Sie können auch angeben, welche Behälter für temporäre Tabellenbereiche für die Datenbanken verwendet werden sollen. Für die Behälter gibt es folgende Möglichkeiten:

- Es sind dieselben Behälter wie die, die für den Katalogknoten der jeweiligen Datenbanken definiert sind (Standardeinstellung).
- Es sind dieselben Behälter wie die, die für eine andere Datenbankpartition definiert sind.
- Sie wurden noch nicht erstellt. Sie müssen jeder Datenbank mit Hilfe der Anweisung ALTER TABLESPACE Tabellenbereichsbehälter für temporäre Tabellen hinzufügen, bevor die Datenbank verwendet werden kann.

Sie können eine Datenbank in der neuen Partition nicht zur Speicherung von Daten verwenden, bevor die neue Datenbankpartition durch eine Änderung nicht in mindestens eine Datenbankpartitionsgruppe aufgenommen wurde.

Sie können nicht von einem Einzelpartitionssystem zu einem Mehrfachpartitionssystem wechseln, indem Sie Ihrem System einfach eine Partition hinzufügen, da für die Umverteilung von Daten zwischen Partitionen für jede betroffene Tabelle ein entsprechender Partitionierungsschlüssel erforderlich ist. Beim Erstellen einer Tabelle in einer Umgebung mit mehreren Partitionen werden die Partitionierungsschlüssel automatisch generiert. In einer Umgebung mit nur einer Partition können Partitionierungsschlüssel mit Hilfe der SQL-Anweisungen CREATE TABLE oder ALTER TABLE explizit erstellt werden.

Anmerkung: Wenn keine Datenbanken im System definiert sind und Sie Enterprise Server Edition auf einem UNIX®-System ausführen, editieren Sie die Datei `db2nodes.cfg`, um eine neue Definition für eine Datenbankpartition hinzuzufügen. Verwenden Sie keine der beschrieben Prozeduren, da sie nur gelten, wenn eine Datenbank vorhanden ist.

Für Windows NT ist zu beachten: Wenn Sie Enterprise Server Edition unter Windows NT verwenden und keine Datenbanken im Exemplar definiert haben, verwenden Sie den Befehl DB2NCRT, um das Datenbanksystem zu skalieren. Wenn Sie jedoch bereits Datenbanken haben, verwenden Sie den Befehl DB2START ADDNODE, um sicherzustellen, dass beim Skalieren des Systems für jede vorhandene Datenbank eine Datenbankpartition erstellt wird. Unter Windows NT sollten Sie die Knotenkonfigurationsdatei (`db2nodes.cfg`) nie manuell editieren, da dies zu Inkonsistenzen in der Datei führen kann.

Zugehörige Tasks:

- „Hinzufügen einer Partition zu einem aktiven Datenbanksystem" auf Seite 333
- „Hinzufügen einer Partition zu einem gestoppten Datenbanksystem unter Windows NT" auf Seite 334
- „Löschen einer Datenbankpartition" auf Seite 339

Hinzufügen einer Partition zu einem aktiven Datenbanksystem

Sie können einem partitionierten Datenbanksystem neue Datenbankpartitionen hinzufügen, während es aktiv ist und Anwendungen mit Datenbanken verbunden sind. Allerdings wird der hinzugefügte Server erst dann für alle Datenbanken verfügbar, wenn der Datenbankmanager gestoppt und erneut gestartet wird.

Vorgehensweise:

Gehen Sie wie folgt vor, um einem aktiven Datenbankmanager eine Datenbankpartition hinzuzufügen:

1. Führen Sie den Befehl DB2START in jeder vorhandenen Datenbankpartition aus.

   Geben Sie auf allen Plattformen die neuen Partitionswerte für die Parameter DBPARTITIONNUM, ADD DBPARTITIONNUM, HOSTNAME, PORT und NETNAME an. Auf der Windows NT-Plattform geben Sie außerdem die Parameter COMPUTER, USER und PASSWORD an.

   Sie können auch die Quelle für die Definitionen von Tabellenbereichsbehältern für temporäre Tabellen angeben, die mit den Datenbanken erstellt werden müs-

sen. Wenn Sie keine Tabellenbereichsinformationen angeben, werden die Definitionen für Tabellenbereichsbehälter für temporäre Tabellen vom Katalogknoten der einzelnen Datenbanken abgerufen.

Wenn der Befehl DB2START ausgeführt wurde, wird der neue Server gestoppt.

2. Stoppen Sie den Datenbankmanager in allen Partitionen durch Ausführen des Befehls DB2STOP.

   Wenn Sie alle Datenbankpartitionen in einem System stoppen, wird die Knotenkonfigurationsdatei aktualisiert und enthält nun die neue Datenbankpartition. Die Knotenkonfigurationsdatei wird erst dann mit den neuen Serverinformationen aktualisiert, wenn DB2STOP ausgeführt wird. Dadurch wird sichergestellt, dass der Befehl ADD DBPARTITIONNUM, der aufgerufen wird, wenn Sie den Parameter ADDNODE im Befehl DB2START angeben, in der richtigen Datenbankpartition ausgeführt wird. Wenn das Dienstprogramm beendet ist, wird die neue Serverpartition gestoppt.

3. Starten Sie den Datenbankmanager durch Ausführen des Befehls DB2START.

   Nun wird die hinzugefügte Datenbankpartition gemeinsam mit dem restlichen System gestartet.

   Wenn alle Datenbankpartitionen des Systems aktiv sind, können Sie Aktionen durchführen, die das gesamte System betreffen, wie zum Beispiel das Erstellen oder Löschen einer Datenbank.

   Anmerkung: Möglicherweise müssen Sie den Befehl DB2START zweimal ausführen, damit alle Datenbankpartitionsserver auf die neue Datei `db2nodes.cfg` zugreifen.

4. Sichern Sie alle Datenbanken in der neuen Datenbankpartition (optional).
5. Verteilen Sie Daten auf die neue Datenbankpartition (optional).

Zugehörige Konzepte:

- „Partitionen in einer partitionierten Datenbank" auf Seite 332

Zugehörige Tasks:

- „Hinzufügen einer Partition zu einem gestoppten Datenbanksystem unter Windows NT" auf Seite 334
- „Hinzufügen einer Partition zu einem gestoppten Datenbanksystem unter UNIX" auf Seite 336

Hinzufügen einer Partition zu einem gestoppten Datenbanksystem unter Windows NT

Sie können einem partitionierten Datenbanksystem neue Datenbankpartitionen hinzufügen, während es gestoppt ist. Die neu hinzugefügte Datenbankpartition wird erst nach einem Neustart des Datenbankmanagers für alle Datenbanken verfügbar.

Voraussetzungen:

Sie müssen den neuen Server installieren, bevor Sie eine Partition auf ihm erstellen können.

Vorgehensweise:

Gehen Sie wie folgt vor, um einem gestoppten partitionierten Datenbankserver eine Partition hinzuzufügen:

1. Setzen Sie den Befehl DB2STOP ab, um alle Datenbankpartitionen zu stoppen.
2. Führen Sie den Befehl ADD DBPARTITIONNUM auf dem neuen Server aus.

   Eine Datenbankpartition wird lokal für jede Datenbank erstellt, die bereits im System vorhanden ist. Die Datenbankparameter für die neuen Datenbankpartitionen werden auf die Standardwerte gesetzt, und jede Datenbankpartition bleibt leer, bis Sie Daten dorthin versetzen. Aktualisieren Sie die Datenbankkonfigurationsparameter, so dass sie mit denen in den anderen Datenbankpartitionen übereinstimmen.
3. Führen Sie den Befehl DB2START aus, um das Datenbanksystem zu starten. Beachten Sie, dass die Knotenkonfigurationsdatei (`db2nodes.cfg`) bereits bei der Installation des neuen Servers aktualisiert wurde, um den neuen Server aufzunehmen.
4. Aktualisieren Sie die Konfigurationsdatei in der neuen Partition wie folgt:
   a. Führen Sie den Befehl DB2START in jeder vorhandenen Datenbankpartition aus.

      Geben Sie die neuen Partitionswerte für die Parameter DBPARTITIONNUM, ADDDB2PARTITIONNUM, HOSTNAME, PORT und NETNAME sowie für die Parameter COMPUTER, USER und PASSWORD an.

      Sie können auch die Quelle für die Definitionen von Tabellenbereichsbehältern für temporäre Tabellen angeben, die mit den Datenbanken erstellt werden müssen. Wenn Sie keine Tabellenbereichsinformationen angeben, werden die Definitionen für Tabellenbereichsbehälter für temporäre Tabellen vom Katalogknoten der einzelnen Datenbanken abgerufen.

      Wenn der Befehl DB2START ausgeführt wurde, wird der neue Server gestoppt.
   b. Stoppen Sie den gesamten Datenbankmanager durch Ausführen des Befehls DB2STOP.

      Wenn Sie alle Datenbankpartitionen in einem System stoppen, wird die Knotenkonfigurationsdatei aktualisiert und enthält nun die neue Datenbankpartition. Die Knotenkonfigurationsdatei wird erst dann mit den neuen Serverinformationen aktualisiert, wenn DB2STOP ausgeführt wird. Dadurch wird sichergestellt, dass der Befehl ADD DB2PARTITIONNUM, der aufgerufen wird, wenn Sie den Parameter ADDDB2PARTITIONNUM im Befehl DB2START angeben, in der richtigen Datenbankpartition ausgeführt wird. Wenn das Dienstprogramm beendet ist, wird die neue Serverpartition gestoppt.
5. Starten Sie den Datenbankmanager durch Ausführen des Befehls DB2START.

   Nun wird die hinzugefügte Datenbankpartition mit dem restlichen System gestartet.

   Wenn alle Datenbankpartitionen des Systems aktiv sind, können Sie Aktionen durchführen, die das gesamte System betreffen, wie zum Beispiel das Erstellen oder Löschen einer Datenbank.

   Anmerkung: Möglicherweise müssen Sie den Befehl DB2START zweimal ausführen, damit alle Datenbankpartitionsserver auf die neue Datei `db2nodes.cfg` zugreifen.
6. Sichern Sie alle Datenbanken in der neuen Datenbankpartition (optional).
7. Verteilen Sie Daten auf die neue Datenbankpartition (optional).

Zugehörige Konzepte:

- „Partitionen in einer partitionierten Datenbank” auf Seite 332
- „Wiederherstellung nach Fehlern beim Hinzufügen von Knoten” auf Seite 338

Zugehörige Tasks:

- „Hinzufügen einer Partition zu einem aktiven Datenbanksystem" auf Seite 333
- „Hinzufügen einer Partition zu einem gestoppten Datenbanksystem unter UNIX" auf Seite 336

Hinzufügen einer Partition zu einem gestoppten Datenbanksystem unter UNIX

Sie können einem partitionierten Datenbanksystem neue Datenbankpartitionen hinzufügen, während es gestoppt ist. Die neu hinzugefügte Datenbankpartition wird erst nach einem Neustart des Datenbankmanagers für alle Datenbanken verfügbar.

Voraussetzungen:

Sie müssen den neuen Server installieren, falls er nicht vorhanden ist. Dies umfasst folgende Aufgaben:

- Einrichten des Zugriffs auf ausführbare Dateien (durch Anhängen gemeinsamer Dateisysteme oder lokale Kopien)
- Synchronisieren von Betriebssystemdateien mit denen auf vorhandenen Prozessoren
- Sicherstellen des Zugriffs auf das Verzeichnis `sqllib` als gemeinsames Dateisystem
- Sicherstellen der richtigen Werte für die relevanten Betriebssystemparameter (z. B. die maximale Anzahl von Prozessen)

Sie müssen auch den Hostnamen auf dem Namensserver oder in der Datei `hosts` im Verzeichnis `etc` in allen Datenbankpartitionen registrieren.

Vorgehensweise:

Gehen Sie wie folgt vor, um einem gestoppten partitionierten Datenbankserver eine Partition hinzuzufügen:

1. Setzen Sie den Befehl DB2STOP ab, um alle Datenbankpartitionen zu stoppen.
2. Führen Sie den Befehl ADD DB2PARTITIONNUM auf dem neuen Server aus.

   Eine Datenbankpartition wird lokal für jede Datenbank erstellt, die bereits im System vorhanden ist. Die Datenbankparameter für die neuen Datenbankpartitionen werden auf die Standardwerte gesetzt, und jede Datenbankpartition bleibt leer, bis Sie Daten dorthin versetzen. Aktualisieren Sie die Datenbankkonfigurationsparameter, so dass sie mit denen in den anderen Datenbankpartitionen übereinstimmen.
3. Führen Sie den Befehl DB2START aus, um das Datenbanksystem zu starten. Beachten Sie, dass die Knotenkonfigurationsdatei (`db2nodes.cfg`) bereits bei der Installation des neuen Servers aktualisiert wurde, um den neuen Server aufzunehmen.
4. Aktualisieren Sie die Konfigurationsdatei in der neuen Partition wie folgt:
   a. Führen Sie den Befehl DB2START in jeder vorhandenen Datenbankpartition aus.

      Geben Sie die neuen Partitionswerte für die Parameter DB2PARTITIONNUM, ADDDB2PARTITIONNUM, HOSTNAME, PORT und NETNAME sowie für die Parameter COMPUTER, USER und PASSWORD an.

Sie können auch die Quelle für die Definitionen von Tabellenbereichsbehältern für temporäre Tabellen angeben, die mit den Datenbanken erstellt werden müssen. Wenn Sie keine Tabellenbereichsinformationen angeben, werden die Definitionen für Tabellenbereichsbehälter für temporäre Tabellen vom Katalogknoten der einzelnen Datenbanken abgerufen.

Wenn der Befehl DB2START ausgeführt wurde, wird der neue Server gestoppt.

b. Stoppen Sie den gesamten Datenbankmanager durch Ausführen des Befehls DB2STOP.

Wenn Sie alle Datenbankpartitionen in einem System stoppen, wird die Knotenkonfigurationsdatei aktualisiert und enthält nun die neue Datenbankpartition. Die Knotenkonfigurationsdatei wird erst dann mit den neuen Serverinformationen aktualisiert, wenn DB2STOP ausgeführt wird. Dadurch wird sichergestellt, dass der Befehl ADD DB2PARTITIONNUM, der aufgerufen wird, wenn Sie den Parameter ADDDB2PARTITIONNUM im Befehl DB2START angeben, in der richtigen Datenbankpartition ausgeführt wird. Wenn das Dienstprogramm beendet ist, wird die neue Serverpartition gestoppt.

5. Starten Sie den Datenbankmanager durch Ausführen des Befehls DB2START.

Nun wird die hinzugefügte Datenbankpartition mit dem restlichen System gestartet.

Wenn alle Datenbankpartitionen des Systems aktiv sind, können Sie Aktionen durchführen, die das gesamte System betreffen, wie zum Beispiel das Erstellen oder Löschen einer Datenbank.

Anmerkung: Möglicherweise müssen Sie den Befehl DB2START zweimal ausführen, damit alle Datenbankpartitionsserver auf die neue Datei `db2nodes.cfg` zugreifen.

6. Sichern Sie alle Datenbanken in der neuen Datenbankpartition (optional).
7. Verteilen Sie Daten auf die neue Datenbankpartition (optional).

Sie können die Konfigurationsdatei auch wie folgt manuell aktualisieren:

1. Editieren Sie die Datei `db2nodes.cfg` und fügen Sie die neue Datenbankpartition ein.
2. Setzen Sie den folgenden Befehl ab, um den neuen Knoten zu starten: `DB2START DB2PARTITIONNUM partitionsnummer`

Geben Sie als Wert für *partitionsnummer* die Nummer an, die Sie der neuen Datenbankpartition zuordnen.

3. Wenn der neue Server eine logische Datenbankpartition sein soll (d. h. nicht Knoten 0), verwenden Sie den Befehl DB2SET, um die Registriervariable DB2PARTITIONNUM zu aktualisieren. Geben Sie die Nummer der Datenbankpartition an, die Sie hinzufügen.
4. Führen Sie den Befehl ADD NODE in der neuen Datenbankpartition aus.

Dieser Befehl erstellt lokal eine Datenbankpartition für jede Datenbank, die bereits im System vorhanden ist. Die Datenbankparameter für die neuen Datenbankpartitionen werden auf die Standardwerte gesetzt, und jede Datenbankpartition bleibt leer, bis Sie Daten dorthin versetzen. Aktualisieren Sie die Datenbankkonfigurationsparameter, so dass sie mit denen in den anderen Datenbankpartitionen übereinstimmen.

5. Wenn der Befehl ADD DB2PARTITIONNUM ausgeführt wurde, setzen Sie den Befehl DB2START ab, um die übrigen Datenbankpartitionen des Systems zu starten.

Führen Sie keine Aktionen aus, die das gesamte System betreffen, wie zum Beispiel das Erstellen oder Löschen einer Datenbank, bis alle Datenbankpartitionen erfolgreich gestartet wurden.

Zugehörige Konzepte:

- „Wiederherstellung nach Fehlern beim Hinzufügen von Knoten" auf Seite 338

Zugehörige Tasks:

- „Hinzufügen einer Partition zu einem aktiven Datenbanksystem" auf Seite 333
- „Hinzufügen einer Partition zu einem gestoppten Datenbanksystem unter Windows NT" auf Seite 334
- „Löschen einer Datenbankpartition" auf Seite 339

Wiederherstellung nach Fehlern beim Hinzufügen von Knoten

Da in Version 8.1 und späteren Versionen DB2® „verdeckte" Pufferpools erstellt, um eine automatische Standardunterstützung für alle Seitengrößen für Pufferpools bereitzustellen, schlägt das Hinzufügen von Knoten nicht deshalb fehl, weil Pufferpools nicht vorhanden sind. Wenn allerdings einer dieser „verdeckten" Pufferpools verwendet wird, kann die Leistung erheblich beeinträchtigt werden, da die verdeckten Pufferpools sehr klein sind. Wenn ein verdeckter Pufferpool verwendet wird, wird eine Nachricht in das Benachrichtigungsprotokoll für die Systemverwaltung geschrieben.

Verdeckte Pufferpools werden in Knotenhinzufügeszenarios unter den folgenden Umständen verwendet:

- Sie fügen Knoten einer partitionierten Datenbank hinzu, die mindestens einen Tabellenbereich für temporäre Systemtabellen mit einer anderen Seitengröße als die Standardgröße von 4 KB besitzt. Wenn ein Knoten erstellt wird, ist nur der Pufferpool IBMDEFAULTDP vorhanden und dieser Pufferpool hat eine Seitengröße von 4 KB.

  Betrachten Sie die folgenden Beispiele:

  1. Sie verwenden den Befehl DB2START, um der aktuellen partitionierten Datenbank einen Knoten hinzuzufügen:

     ```
     DB2START DB2PARTITIONNUM 2 ADD DB2PARTITIONNUM HOSTNAME neuerhost PORT 2
     ```

  2. Sie verwenden den Befehl ADD DB2PARTITIONNUM, nachdem Sie die Datei `db2nodes.cfg` manuell mit der neuen Knotenbeschreibung aktualisiert haben.

  Eine Möglichkeit zur Vermeidung dieser Probleme ist die Angabe der Klausel WITHOUT TABLESPACES im Befehl ADD NODE oder **db2start**. Anschließend müssen Sie die Pufferpools mit der Anweisung CREATE BUFFERPOOL erstellen und dem Pufferpool die Tabellenbereiche für temporäre Systemtabellen mit der Anweisung ALTER TABLESPACE zuordnen.
- Sie fügen Knoten einer vorhandenen Datenbankpartitionsgruppe hinzu, die mindestens einen Tabellenbereich mit einer anderen Seitengröße als der Standardseitengröße von 4 KB hat. Dies geschieht, weil die auf dem neuen Knoten erstellten Pufferpools mit einer vom Standard abweichenden Seitengröße für die Tabellenbereiche nicht aktiviert wurden.

  Anmerkung: In früheren Versionen von DB2 arbeitete dieser Befehl mit dem Schlüsselwort NODEGROUP anstelle der Schlüsselwörter DATABASE PARTITION GROUP.

Betrachten Sie das folgende Beispiel:

– Sie verwenden die Anweisung ALTER DATABASE PARTITION GROUP, um einer Datenbankpartitionsgruppe wie folgt einen Knoten hinzuzufügen:

```
DB2START
CONNECT TO mpp1
ALTER DATABASE PARTITION GROUP ng1 ADD NODE (2)
```

Eine Möglichkeit, dieses Problems zu vermeiden, besteht darin, Pufferpools für jede Seitengröße zu erstellen und die Verbindung zur Datenbank wieder herzustellen, bevor Sie die folgende Anweisung ALTER DATABASE PARTITION GROUP absetzen:

```
DB2START
CONNECT TO mpp1
CREATE BUFFERPOOL bp1 SIZE 1000 PAGESIZE 8192
CONNECT RESET
CONNECT TO mpp1
ALTER DATABASE PARTITION GROUP ng1 ADD NODE (2)
```

Anmerkung: Wenn die Datenbankpartitionsgruppe Tabellenbereiche mit der Standardseitengröße hat, wird die folgende Nachricht zurückgegeben:

```
SQL1759W Es ist erforderlich, die Knotengruppe umzuverteilen,
      um die Datenpartitionierung für Objekte in der Knotengruppe
      "ng1" zu ändern, damit hinzugefügte Knoten aufgenommen
      oder gelöschte Knoten entfernt werden können.
```

Zugehörige Tasks:

- „Hinzufügen einer Partition zu einem aktiven Datenbanksystem" auf Seite 333
- „Hinzufügen einer Partition zu einem gestoppten Datenbanksystem unter Windows NT" auf Seite 334

Löschen einer Datenbankpartition

Sie können eine Datenbankpartition löschen, die von keiner Datenbank genutzt wird, und den Computer für andere Zwecke freistellen.

Voraussetzungen:

Prüfen Sie, dass die Partition nicht in Gebrauch ist, indem Sie den Befehl DROP NODE VERIFY oder die API-Funktion *sqledrpn* absetzen.

- Wenn Sie die Nachricht SQL6034W (Der Knoten wird von keiner Datenbank verwendet) empfangen, können Sie die Partition löschen.
- Wenn Sie die Nachricht SQL6035W (Knoten wird von Datenbank verwendet) empfangen, verwenden Sie den Befehl REDISTRIBUTE NODEGROUP, um die Daten von der Datenbankpartition, die Sie löschen wollen, auf andere Datenbankpartitionen vom Aliasnamen der Datenbank umzuverteilen.

Stellen Sie außerdem sicher, dass alle Transaktionen, die von der betreffenden Datenbankpartition koordiniert wurden, erfolgreich festgeschrieben oder rückgängig gemacht wurden. Dazu ist möglicherweise eine Wiederherstellung nach einem Systemabsturz auf anderen Servern erforderlich. Wenn Sie zum Beispiel die Koordinatordatenbankpartition (d. h. den Koordinatorknoten) löschen und vor dem Löschen des Koordinatorknotens eine andere an einer Transaktion beteiligte Datenbankpartition abgestürzt war, ist die abgestürzte Datenbankpartition nicht in der Lage, den Koordinatorknoten nach dem Ergebnis unbestätigter Transaktionen abzufragen.

Vorgehensweise:

Gehen Sie wie folgt vor, um eine Datenbankpartition zu löschen:

1. Setzen Sie den Befehl DB2STOP mit dem Parameter DROP NODENUM ab, um die Datenbankpartition zu löschen. Nach erfolgreicher Ausführung des Befehls wird das System gestoppt.
2. Starten Sie den Datenbankmanager mit dem Befehl DB2START.

Zugehörige Konzepte:

- „Verwaltung der Datenbankserverkapazität" auf Seite 331
- „Partitionen in einer partitionierten Datenbank" auf Seite 332

Kapitel 11. Umverteilen von Daten auf Datenbankpartitionen

Dieses Kapitel enthält Informationen zur Ermittlung des Zeitpunkts, zu dem eine Umverteilung von Daten auf Partitionen erforderlich ist, zur Durchführung der Umverteilung sowie zur Wiederherstellung nach Fehlern bei der Umverteilung.

Umverteilung von Daten

Zur Umverteilung von Tabellendaten auf die Partitionen in einer partitionierten Datenbank verwenden Sie den Befehl REDISTRIBUTE DATABASE PARTITION GROUP.

Anmerkung: In früheren Versionen von DB2® arbeitete dieser Befehl mit dem Schlüsselwort NODEGROUP anstelle der Schlüsselwörter DATABASE PARTITION GROUP.

In einer partitionierten Datenbank kann die Umverteilung der Daten folgenden Zwecken dienen:

- Neuausgleich von Datenvolumen und verteilte Verarbeitung von Auslastungen in allen Datenbankpartitionen

  Wenn der Datenzugriff auf mehrere Partitionen verteilt werden kann, verbessert sich die Leistung.
- Ungleichmäßige (gewichtete) Verteilung der Daten auf die Datenbankpartitionen

  Zugriffsleistung und Durchsatz lassen sich möglicherweise erhöhen, wenn Sie die Daten in einer Tabelle, auf die häufig zugegriffen wird, so umverteilen, dass die Daten, auf die selten zugegriffen wird, in einer kleinen Anzahl von Datenbankpartitionen in der Datenbankpartitionsgruppe gespeichert werden, während die Daten, auf die häufig zugegriffen wird, auf eine größere Anzahl von Partitionen verteilt werden. Dadurch würden die Zugriffsleistung und der Durchsatz für die am häufigsten ausgeführten Anwendungen verbessert.

Verwenden Sie zur Beibehaltung der Tabellenkollokation den Befehl REDISTRIBUTE DATABASE PARTITION GROUP, um Daten auf der Ebene der Datenbankpartitionsgruppe umzuverteilen. Alle Tabellen werden in einer einzigen Operation umverteilt. Zur Umsetzung der angegebenen Datenverteilung teilt der Befehl REDISTRIBUTE DATABASE PARTITION GROUP Tabellen beim Versetzen der Zeilen auf die Datenbankpartitionen auf. Abhängig von der angegebenen Option kann das Dienstprogramm entweder eine Zielpartitionierungszuordnung generieren oder eine vorhandene Partitionierungszuordnung als Eingabe verwenden.

Funktionsweise der Umverteilung von Daten auf Datenbankpartitionen

Eine Datenumverteilung wird für eine Gruppe von Tabellen in der angegebenen Datenbankpartitionsgruppe einer Datenbank durchgeführt. Sie müssen eine Verbindung zu der Datenbank in der Katalogdatenbankpartition herstellen, bevor Sie den Befehl REDISTRIBUTE DATABASE PARTITION GROUP ausführen, um das Dienstprogramm zur Datenumverteilung (Data Redistribution) aufzurufen. Das Dienstprogramm verwendet die Quellenpartitionierungszuordnung und die Zielpartitionierungszuordnung, um zu ermitteln, welche Hashpartitionen einer neuen Speicherposition, d. h. einer neuen Datenbankpartitionsnummer, zugeordnet wurden. Alle Zeilen, die zu einer Partition mit einer neuen Speicherposition gehören,

werden von der in der Quellenpartitionierungszuordnung angegebenen Datenbankpartition in die in der Zielpartitionierungszuordnung angegebenen Datenbankpartition versetzt.

Das Dienstprogramm zur Datenumverteilung führt die folgenden Schritte aus:

1. Es empfängt eine neue Partitionierungszuordnungs-ID für die Zielpartitionierungszuordnung und fügt diese in die Katalogsicht SYSCAT.PARTITIONMAPS ein.
2. Es aktualisiert die Spalte REBALANCE_PMAP_ID in der Katalogsicht SYSCAT.DBPARTITIONGROUPS für die Datenbankpartitionsgruppe mit der neuen Partitionierungszuordnungs-ID.
3. Es fügt die neuen Datenbankpartitionen der Katalogsicht SYSCAT.DBPARTITIONGROUPDEF hinzu.
4. Es setzt den Wert der Spalte IN_USE in der Katalogsicht SYSCAT.DBPARTITIONGROUPDEF auf 'D' (DROP) für jede Datenbankpartition, die zu löschen ist.
5. Es führt eine COMMIT-Operation für die Katalogaktualisierungen aus.
6. Es erstellt Datenbankdateien für alle neuen Datenbankpartitionen.
7. Es verteilt die Daten tabellenweise für jede Tabelle in der Datenbankpartitionsgruppe um.
   a. Es sperrt die Zeile für die Tabelle in der Katalogtabelle SYSTABLES.
   b. Es macht alle Pakete, die mit dieser Tabelle arbeiten, ungültig. Die der Tabelle zugeordnete Partitionierungszuordnungs-ID wird geändert, da die Tabellenzeilen neu verteilt werden. Da die Pakete ungültig gemacht werden, muss der Compiler die neuen Partitionierungsinformationen für die Tabelle abrufen und entsprechende Pakete generieren.
   c. Es sperrt die Tabelle im Exklusivmodus (Exclusive).
   d. Es verwendet DELETE- und INSERT-Anweisungen, um die Daten in der Tabelle umzuverteilen.
   e. Wenn die Umverteilungsoperation erfolgreich abgeschlossen wird, setzt es eine COMMIT-Anweisung für die Tabelle ab und fährt mit der nächsten Tabelle in der Datenbankpartitionsgruppe fort. Falls die Operation fehlschlägt, bevor die Tabelle vollständig umverteilt ist, macht das Dienstprogramm die Aktualisierungen an der Tabelle rückgängig (ROLLBACK), beendet die gesamte Umverteilungsoperation und gibt eine Fehlernachricht zurück.
8. Es löscht Datenbankdateien und Einträge in der Katalogsicht SYSCAT.NODEGROUPDEF für Datenbankpartitionen, die zuvor als zu löschen ('D') markiert wurden.
9. Es aktualisiert den Datensatz für die Datenbankpartitionsgruppe in der Katalogsicht SYSCAT.NODEGROUPS, um den Wert von PMAP_ID auf den Wert von REBALANCE_PMAP_ID und den Wert von REBALANCE_PMAP_ID auf NULL zu setzen.
10. Es löscht die alte Partitionierungszuordnung aus der Katalogsicht SYSCAT.PARTITIONMAPS.
11. Es führt eine COMMIT-Operation für alle Änderungen aus.

Zugehörige Konzepte:

- „Protokollspeicheranforderungen für die Datenumverteilung" auf Seite 346
- „Fehlerbehebung nach einer Umverteilung" auf Seite 347

Zugehörige Tasks:

- „Umverteilen von Daten auf Partitionen" auf Seite 343
- „Feststellen der Erforderlichkeit einer Datenumverteilung" auf Seite 343

Feststellen der Erforderlichkeit einer Datenumverteilung

Vor der Entscheidung zur Umverteilung von Daten müssen Sie herausfinden, ob die Daten unter den Partitionen ungleich verteilt sind. Wenn Sie die Informationen über die aktuelle Verteilung haben, können Sie mit Hilfe dieser Informationen eine eigene Umverteilungsdatei oder Partitionierungszuordnung erstellen.

Vorgehensweise:

Gehen Sie wie folgt vor, um Informationen über die aktuellen Datenverteilungen für Partitionen in einer Datenbankpartitionsgruppe abzurufen:

1. Ermitteln Sie, ob Datenbankpartitionen ungleiche Verteilungen von Zeilen aufweisen.

   Verwenden Sie für die größte Tabelle eine geeignete Partitionierungsspalte und geben Sie ein Abfrage wie die folgende ein:

   ```
   SELECT PARTITION(spaltenname), COUNT(*) FROM tabellenname
      GROUP BY PARTITION(spaltenname)
      ORDER BY PARTITION(spaltenname) DESC
      FETCH FIRST 100 ROWS ONLY
   ```

   Die SQL-Funktionen PARTITION und DBPARTITIONNUM ermitteln die aktuelle Datenverteilung über Hashpartitionen oder Datenbankpartitionen. Die Funktion PARTITION gibt den Partitionierungszuordnungsindex für jede Zeile der Tabelle zurück. Die Funktion DBPARTITIONNUM liefert die Partitionsnummer der Zeile.
2. Führen Sie diese Abfrage für andere große Tabellen aus, die über die Datenbankpartitionsgruppe partitioniert sind.
3. Verwenden Sie die Informationen, um eine Verteilungsdatei und eine Zielpartitionierungszuordnung zu erstellen.

Anmerkung: Außerdem haben Sie die Möglichkeit, das Dienstprogramm `AutoLoader` für automatisches Laden mit der Option ANALYZE zu verwenden, um eine Datenverteilungsdatei zu erstellen. Sie können diese Datei als Eingabe für das Dienstprogramm zur Datenumverteilung verwenden.

Zugehörige Konzepte:

- „Umverteilung von Daten" auf Seite 341
- „Protokollspeicheranforderungen für die Datenumverteilung" auf Seite 346

Zugehörige Tasks:

- „Umverteilen von Daten auf Partitionen" auf Seite 343

Umverteilen von Daten auf Partitionen

In einer partitionierten Datenbank können Sie Daten in folgenden Fällen auf die Partitionen zum Ausgleichen des Datenzugriffs umverteilen:

- Wenn einige Partitionen mehr Daten als andere enthalten.
- Wenn auf einige Partitionen häufiger als auf andere zugegriffen wird.

Voraussetzungen:

Protokolldateigröße: Stellen Sie sicher, dass die Protokolldateien für die Operation der Datenumverteilung ausreichend groß sind. Die Protokolldateien der einzelnen betroffenen Partitionen müssen die dort durchgeführten INSERT- und DELETE-Operationen aufnehmen können.

Replizierte gespeicherte Abfragetabellen: Wenn die Daten in einer Datenbankpartitionsgruppe replizierte gespeicherte Abfragetabellen enthält, müssen Sie diese Tabellen löschen, bevor Sie die Daten umverteilen. Nach der Umverteilung der Daten können Sie die gespeicherten Abfragetabellen erneut erstellen.

Einschränkungen:

Die folgenden Operationen können an Objekten der Datenbankpartitionsgruppe ausgeführt werden, während das Dienstprogramm aktiv ist. Sie können jedoch nicht an der Tabelle ausgeführt werden, die momentan umverteilt wird. Mögliche Operationen:

- Erstellen von Indizes für andere Tabellen. Die Anweisung CREATE INDEX verwendet die Partitionierungszuordnung der betroffenen Tabelle.
- Löschen anderer Tabellen. Die Anweisung DROP TABLE verwendet die Partitionierungszuordnung der betroffenen Tabelle.
- Löschen von Indizes für andere Tabellen. Die Anweisung DROP INDEX verwendet die Partitionierungszuordnung der betroffenen Tabelle.
- Abfragen anderer Tabellen
- Aktualisieren anderer Tabellen
- Erstellen neuer Tabellen in einem in der Datenbankpartitionsgruppe definierten Tabellenbereich. Die Anweisung CREATE TABLE verwendet die Zielpartitionierungszuordnung.
- Erstellen von Tabellenbereichen in der Datenbankpartitionsgruppe

Folgende Operationen stehen während der Ausführung des Dienstprogramms nicht zur Verfügung:

- Starten einer weiteren Umverteilungsoperation für die Datenbankpartitionsgruppe
- Ausführen einer Anweisung ALTER TABLE für eine Tabelle in der Datenbankpartitionsgruppe
- Löschen (DROP) der Datenbankpartitionsgruppe
- Ändern der Datenbankpartitionsgruppe

Mit dieser Prozedur können keine Daten umverteilt werden, nachdem eine Partition einem Einzelpartitionssystem hinzugefügt wurde, sofern nicht alle betroffenen Tabellen über einen Partitionierungsschlüssel verfügen. Der Befehl REDISTRIBUTE DATABASE PARTITION GROUP hängt bei der Umverteilung von Daten von Partitionierungsschlüsseln ab. Der Partitionierungsschlüssel wird beim Erstellen einer Tabelle in einer Datenbankpartitionsgruppe mit mehreren Partitionen automatisch generiert oder kann mit Hilfe der SQL-Anweisungen CREATE TABLE oder ALTER TABLE explizit definiert werden. Wurden Ihre Tabellen in einer Partitionsgruppe mit nur einer Partition erstellt und Sie haben in der SQL-Anweisung CREATE TABLE nicht den Partitionierungsschlüssel definiert, liegen keine definierten Partitionierungsschlüssel vor. Vor dem Umverteilen der Daten müssen Sie für jede betroffene Tabelle mit Hilfe der SQL-Anweisung ALTER TABLE einen Partitionsschlüssel erstellen.

Vorgehensweise:

Gehen Sie wie folgt vor, um Daten auf Partitionen in einer Datenbankpartitionsgruppe umzuverteilen:

1. Stellen Sie eine Verbindung zu der Datenbankpartition her, in der die Systemkatalogtabellen enthalten sind.
2. Führen Sie die als Voraussetzungen aufgeführten Operationen durch, falls erforderlich.
3. Setzen Sie den Befehl REDISTRIBUTE DATABASE PARTITION GROUP ab.

   Anmerkung: In früheren Versionen von DB2 arbeitete dieser Befehl mit dem Schlüsselwort NODEGROUP anstelle der Schlüsselwörter DATABASE PARTITION GROUP.

   Geben Sie die folgenden Argumente an:

   Name der Datenbankpartitionsgruppe
   Sie müssen die Datenbankpartitionsgruppe angeben, in der die Daten umzuverteilen sind.

   UNIFORM
   Wenn die Daten gleichmäßig verteilt sind und gleichmäßig verteilt bleiben sollen, geben Sie entweder das Argument UNIFORM an oder übergehen alle Argumente für den Verteilungstyp. UNIFORM ist der Standardwert.

   USING DISTFILE verteilungsdateiname
   Wenn Sie eine benutzerdefinierte Verteilung angeben wollen, die eine ungleichmäßige Datenverteilung korrigiert oder herbeiführt, geben Sie den Namen der Verteilungsdatei an. Das Dienstprogramm zur Datenumverteilung verwendet diese Datei zum Aufbau einer Zielpartitionierungszuordnung.

   USING TARGETMAP zielzuordnungsname
   Das Dienstprogramm zur Datenumverteilung verwendet die angegebene Zielzuordnung direkt.

   Weitere Einzelheiten enthalten die Informationen zum Befehlszeilendienstprogramm REDISTRIBUTE DATABASE PARTITION GROUP.
4. Gehen Sie nach Abschluss der Umverteilung wie folgt vor:
   - Erstellen Sie alle replizierten gespeicherten Abfragetabellen erneut, die Sie vor der Umverteilung gelöscht haben.
   - Führen Sie den Befehl RUNSTATS aus, um Statistikdaten über die Datenverteilung zu sammeln, die vom SQL-Compiler und Optimierungsprogramm bei der Auswahl von Datenzugriffsplänen für Abfragen verwendet werden können.

Anmerkung: Die EXPLAIN-Tabellen enthalten Informationen über die Partitionierungszuordnung, die zur Umverteilung der Daten verwendet wird.

Zugehörige Konzepte:

- „Umverteilung von Daten" auf Seite 341
- „Protokollspeicheranforderungen für die Datenumverteilung" auf Seite 346
- „Fehlerbehebung nach einer Umverteilung" auf Seite 347

Zugehörige Tasks:

- „Feststellen der Erforderlichkeit einer Datenumverteilung" auf Seite 343

Protokollspeicheranforderungen für die Datenumverteilung

Bevor Sie Daten über Partitionen umverteilen, müssen Sie die Protokollspeicheranforderungen in die Überlegungen mit einbeziehen.

Die Größe des Protokolls muss ausreichend sein, damit die INSERT- und DELETE-Operationen darin aufgenommen werden können, die in jeder von der Umverteilung betroffenen Datenbankpartition ausgeführt werden. Die größten Protokollanforderungen betreffen entweder die Datenbankpartition, die die meisten Daten abgibt, oder die Datenbankpartition, die die meisten Daten dazuerhält. Wenn Sie einen Wechsel zu einer größeren Zahl von Datenbankpartitionen vornehmen, können Sie anhand des Verhältnisses der aktuellen Anzahl von Datenbankpartitionen zur neuen Anzahl von Datenbankpartitionen die Anzahl der INSERT- und DELETE-Operationen abschätzen. Betrachten Sie zum Beispiel eine Umverteilung von Daten, die vor der Umverteilung gleichmäßig verteilt sind. Wenn Sie die Anzahl der Datenbankpartitionen von vier auf fünf erhöhen, werden ca. 20 Prozent der Daten der vier ursprünglichen Datenbankpartitionen in die neue Datenbankpartition verschoben. Dies bedeutet, dass 20 Prozent der DELETE-Operationen in jeder der vier ursprünglichen Datenbankpartitionen und sämtliche INSERT-Operationen in der neuen Datenbankpartition stattfinden.

Betrachten Sie demgegenüber eine nicht gleichmäßige Verteilung von Daten, wie sie zum Beispiel vorliegt, wenn der Partitionierungsschlüssel viele NULL-Werte enthält. In diesem Fall würden alle Zeilen, die einen NULL-Wert im Partitionierungsschlüssel enthalten, aus einer Datenbankpartition unter dem alten Partitionierungsschema in eine andere Datenbankpartition unter dem neuen Partitionierungsschema versetzt. Infolgedessen erhöht sich der erforderliche Protokollspeicherbereich in diesen beiden Datenbankpartitionen und kann möglicherweise den unter der Annahme einer gleichmäßigen Verteilung berechneten Wert übersteigen.

Die Umverteilung der Daten jeder einzelnen Tabelle stellt nur eine Transaktion dar. Aus diesem Grund müssen Sie bei der Abschätzung des Protokollspeicherbereichs den Prozentsatz an Änderungen, wie die 20 Prozent in obigem Beispiel, mit der Größe der größten Tabelle multiplizieren. Beachten Sie jedoch, dass die größte Tabelle vielleicht gleichmäßig verteilt ist, die zweitgrößte Tabelle hingegen zum Beispiel eine oder mehrere unverhältnismäßig große Datenbankpartitionen haben kann. In einem solchen Fall wäre die Verwendung der ungleichmäßig verteilten Tabelle anstelle der größten Tabelle in Betracht zu ziehen.

Anmerkung: Wenn Sie einen Schätzwert für die Höchstmenge an Daten berechnet haben, die in einer Datenbankpartition einzufügen und zu löschen sind, verdoppeln Sie diesen Schätzwert, um den Spitzenwert für die Größe der aktiven Protokolldatei zu ermitteln. Wenn dieser Wert die Begrenzung von 256 GB für die aktive Protokolldatei überschreitet, muss die Datenumverteilung schrittweise erfolgen. Verwenden Sie das Dienstprogramm „makepmap", um eine Reihe von Zielpartitionierungszuordnungen, d. h. eine für jeden Schritt, zu generieren. Darüber hinaus könnten Sie auch den Datenbankkonfigurationsparameter *logsecond* auf den Wert -1 setzen, um die meisten Protokollspeicherprobleme zu vermeiden.

Zugehörige Konzepte:

- „Umverteilung von Daten" auf Seite 341

Fehlerbehebung nach einer Umverteilung

Nach dem Starten der Umverteilungsoperation wird eine Datei in das Unterverzeichnis `redist` des Verzeichnisses `sqllib` geschrieben. In dieser Statusdatei werden alle Operationen, die in Datenbankpartitionen ausgeführt werden, die Namen der Tabellen, die umverteilt wurden, und der Beendigungsstatus der Operation aufgelistet. Wenn eine Tabelle nicht umverteilt werden kann, wird der Name der Tabelle und der betreffende SQLCODE in der Datei aufgeführt. Wenn die Umverteilungsoperation aufgrund eines falschen Eingabeparameters nicht beginnen kann, wird die Datei nicht geschrieben und ein SQLCODE zurückgegeben.

Für die Datei gilt folgende Namenskonvention:

```
Für UNIX®-Plattformen:
  Datenbankname.Datenbankpartitionsgruppenname.zeitmarke
Für andere Plattformen (nicht UNIX):
  Datenbankname\Datenbankpartitionsgruppenname\datum\uhrzeit
```

Anmerkung: Auf Nicht-UNIX-Plattformen werden nur die ersten acht (8) Byte des Datenbankpartitionsgruppennamens verwendet.

Wenn die Operation zur Datenumverteilung fehlschlägt, sind die Daten einiger Tabellen eventuell bereits neu verteilt, während andere Tabellen noch nicht bearbeitet wurden. Dies hat seine Ursache darin, dass die Datenumverteilung jeweils für eine Tabelle gleichzeitig durchgeführt wird. Sie haben zwei Möglichkeiten der Fehlerbehandlung:

- Verwenden der Option CONTINUE, um die Operation zur Datenumverteilung für die übrigen Tabellen fortzusetzen.
- Verwenden der Option ROLLBACK, um die Datenumverteilung rückgängig zu machen und die umverteilten Tabellen in ihren ursprünglichen Zustand zurückzuversetzen. Die ROLLBACK-Operation kann dabei ebenso lange dauern wie die ursprüngliche Umverteilungsoperation.

Bevor Sie eine dieser Optionen verwenden können, muss zuvor eine Operation zur Datenumverteilung fehlgeschlagen sein, so dass die Spalte REBALANCE_PMID in der Tabelle SYSCAT.DBPARTITIONGROUPS auf einen Nicht-NULL-Wert gesetzt ist.

Falls Sie die Statusdatei versehentlich löschen, haben Sie immer noch die Möglichkeit, eine CONTINUE-Operation auszuführen.

Zugehörige Konzepte:

- „Umverteilung von Daten" auf Seite 341

Zugehörige Tasks:

- „Umverteilen von Daten auf Partitionen" auf Seite 343

Gespeicherte Prozeduren und Funktionen zum Umverteilen

Die gespeicherten Prozeduren für eine schrittweise durchgeführte Umverteilung können zur sicheren Umverteilung der Daten einer Datenbankpartitionsgruppe in einer Reihe von Schritten verwendet werden:

1. Analysieren der Datenbankpartition im Hinblick auf Protokollspeicherverfügbarkeit und Abweichung der Datenverteilung
2. Erstellen einer Datenverteilungsdatei für eine bestimmte Tabelle

3. Erstellen und Berichten des Inhalts eines schrittweise durchgeführten Umverteilungsplans für die Datenbankpartitionsgruppe
4. Umverteilen der Daten der Datenbankpartitionsgruppe entsprechend dem erstellten Plan

Bei der Arbeit mit einzelnen Parametern in den folgenden Prozeduren wird der Wert „-1" als Ausgabewert für Parameter verwendet, wenn ihre Werte nicht abgerufen werden können.

Anmerkung: Die gespeicherten Prozeduren und Funktionen zur Umverteilung von Daten funktionieren nur in partitionierten Datenbanken, in denen ein Partitionierungsschlüssel für jede Tabelle definiert ist.

Gespeicherte Prozedur get_swrd_settings

Die Funktion get_swrd_settings liest die vorhandenen Umverteilungsregistrierdatensätze für die angegebene Datenbankpartitionsgruppe.

Tabelle 31. get_swrd_settings - Eingabeparameter

Name	Datentyp	Beschreibung
dbpgName	VARCHAR(128)	Der Name der Datenbankpartitionsgruppe, für die der Umverteilungsprozess ausgeführt werden soll
matchingSpec	SMALLINT	Die bitweisen Feldkennungen aus Tabelle 32, die die Zielfelder angeben, die durch die Ausgabeparameter in Tabelle 33 auf Seite 349 zurückgegeben werden sollen. Die Ausgabeparameter, die nicht erforderlich sind, können auf null gesetzt werden. Wenn zum Beispiel matchingSpec auf den Wert 96 gesetzt wird, was dem Integerwert von (REDIST_STAGE_SIZE | REDIST_NEXT_STEP) entspricht, muss das aufrufende Programm dieser Funktion nur die Parameter stageSize und nextStep angeben, während der Rest der Ausgabeparameter auf null gesetzt werden kann.

Tabelle 32. Bitweise Feldkennungen

Feldname	Hexadezimalwert	Dezimalwert
REDIST_METHOD	0x0001<<0	1
REDIST_PMAP_FILE	0x0001<<1	2
REDIST_DIST_FILE	0x0001<<2	4
REDIST_STEP_SIZE	0x0001<<3	8
REDIST_NUM_STEPS	0x0001<<4	16
REDIST_STAGE_SIZE	0x0001<<5	32
REDIST_NEXT_STEP	0x0001<<6	64
REDIST_PROCESS_STATE	0x0001<<7	128
REDIST_PWEIGHT_START_NODE	0x0001<<8	256
REDIST_PWEIGHT	0x0001<<9	512

Tabelle 33. get_swrd_settings - Ausgabeparameter

Name	Datentyp	Beschreibung
redistMethod	SMALLINT	Die Nummer, die angibt, dass die Umverteilung mit Hilfe der Verteilungsdatei oder der Zielpartitionierungszuordnung auszuführen ist
pMapFile	VARCHAR (255)	Der vollständige Pfad- und Dateiname der Zielpartitionierungszuordnung
distFile	VARCHAR (255)	Der vollständige Pfad- und Dateiname der Datenverteilungsdatei
stepSize	BIGINT	Die maximale Anzahl von Zeilen, die versetzt werden können, bevor eine Festschreibung (COMMIT) aufgerufen werden muss, um einen Fehler wegen eines gefüllten Protokolls zu vermeiden. Diese Anzahl Zeilen kann in jedem Umverteilungsschritt versetzt werden.
totalSteps	SMALLINT	Die Anzahl von Schritten, die zur vollständigen Umverteilung der Daten der angegebenen Datenbankpartitionsgruppe benötigt werden
stageSize	SMALLINT	Die Anzahl von Schritten, die hintereinander ausgeführt werden sollen
nextStep	SMALLINT	Der Index, der die Trennung zwischen den Schritten, die ausgeführt wurden, und den Schritten, die noch auszuführen sind, angibt
processState	SMALLINT	Eine Markierung, die vom Benutzer gesetzt werden kann, um die Umverteilungsphase beim nächsten Schritt (nextStep) zu stoppen
pNumber	VARCHAR(6000)	Eine vorab zugeordnete Zeichenfolge, die entsprechend der Partitionsgewichtung mit allen Partitionsnummern in der Datenbankpartitionsgruppe gefüllt wird. Die Partitionsnummern in der Zeichenfolge werden durch ein Komma („,“) getrennt.
pWeight	VARCHAR(6000)	Eine vorab zugeordnete Zeichenfolge, die mit allen relativen Gewichtungen des Datenträgers in jeder Partition gefüllt wird, die durch die gespeicherte Prozedur SET_SWRD_SETTINGS angegeben wurde. Die Partitionsgewichtungen in der Zeichenfolge werden durch ein Komma („,“) getrennt.

Gespeicherte Prozedur set_swrd_settings

Die Funktion set_swrd_settings erstellt oder ändert die Umverteilungsregistrierdatenbank. Wenn die Registrierdatenbank nicht vorhanden ist, erstellt die Funktion sie und fügt ihr Datensätze hinzu. Wenn die Registrierdatenbank bereits vorhanden ist, stellt die Funktion über das Feld overwriteSpec fest, welche Feldwerte überschrieben werden müssen. Das Feld overwriteSpec gibt dieser Funktion die Möglichkeit, NULL-Eingaben für Felder zu akzeptieren, die nicht aktualisiert werden müssen.

Der Wert „-2“ kann für stepSize und totalSteps in dieser Prozedur verwendet werden, um anzugeben, dass die Anzahl unbegrenzt ist.

Tabelle 34. set_swrd_settings - Eingabeparameter

Name	Datentyp	Beschreibung
dbpgName	VARCHAR(128)	Der Name der Datenbankpartitionsgruppe, für die der Umverteilungsprozess ausgeführt werden soll
overwriteSpec	SMALLINT	Die bitweisen Feldkennungen aus Tabelle 32 auf Seite 348, die die Zielfelder angeben, die in der Registrierdatenbank der Umverteilungseinstellungen zu schreiben oder zu überschreiben sind.

Tabelle 34. set_swrd_settings - Eingabeparameter (Forts.)

Name	Datentyp	Beschreibung
redistMethod	SMALLINT	Die Nummer, die angibt, dass die Umverteilung mit Hilfe der Verteilungsdatei oder der Zielpartitionierungszuordnung auszuführen ist
pMapFile	VARCHAR (255)	Der vollständige Pfad- und Dateiname der Zielpartitionierungszuordnung
distFile	VARCHAR (255)	Der vollständige Pfad- und Dateiname der Datenverteilungsdatei
stepSize	BIGINT	Die maximale Anzahl von Zeilen, die versetzt werden können, bevor eine Festschreibung (COMMIT) aufgerufen werden muss, um einen Fehler wegen eines gefüllten Protokolls zu vermeiden. Diese Anzahl Zeilen kann in jedem Umverteilungsschritt versetzt werden.
totalSteps	SMALLINT	Die Anzahl von Schritten, die zur vollständigen Umverteilung der Daten der angegebenen Datenbankpartitionsgruppe benötigt werden
stageSize	SMALLINT	Die Anzahl von Schritten, die hintereinander ausgeführt werden sollen
nextStep	SMALLINT	Der Index, der die Trennung zwischen den Schritten, die ausgeführt wurden, und den Schritten, die noch auszuführen sind, angibt
processState	SMALLINT	Eine Markierung, die vom Benutzer gesetzt werden kann, um die Umverteilungsphase beim nächsten Schritt (nextStep) zu stoppen
pNumber	VARCHAR(6000)	Eine Zeichenfolge, die alle Partitionsnummern enthält, die der Partitionsgewichtung entsprechen. Jede Partitionsnummer liegt zwischen 0 und 999, und die Nummern werden durch Kommas („,“) getrennt. Leerzeichen sind in der Zeichenfolge nicht zulässig.
pWeight	VARCHAR(6000)	Eine Zeichenfolge, die alle Partitionsgewichtungen enthält, die der Benutzer entsprechend den Partitionsnummern in der Zeichenfolge pNumber angegeben hat. Jede Partitionsgewichtung ist eine Zahl zwischen 0 und SQL_MAXSMALLVAL, und die Zahlen werden durch Kommas („,“) getrennt. Leerzeichen sind in der Zeichenfolge nicht zulässig.

Gespeicherte Prozedur analyze_log_space

Die Funktion analyze_log_space liefert eine Ergebnismenge (einen geöffneten Cursor) von Ergebnissen einer Speicherplatzanalyse, die die folgenden Felder für jede Datenbankpartition der angegebenen Datenbankpartitionsgruppe enthalten.

Tabelle 35. Felder der Funktion analyze_log_space

Spaltenname	Spaltentyp	Beschreibung
PARTITION_NUM	SMALLINT	Die Partitionsnummer der Protokollspeicheranalyse
TOTAL_LOG_SIZE	BIGINT	Der gesamte zugeordnete Protokollspeicher (in Byte). Der Wert -1 gibt eine unbeschränkte Größe an.
AVAIL_LOG_SPACE	BIGINT	Die Größe (in Byte) des Protokollspeichers, die frei ist und durch den Umverteilungsprozess genutzt werden kann
DATA_SKEW	BIGINT	Der absolute Wert (in Byte) für die Größe der Daten, die von der Zieldatengröße abweichen
REQ_LOG_SPACE	BIGINT	Die Größe des Speicherbereichs (in Byte), die zur Erreichung der gewünschten Datenverteilung erforderlich ist
NUM_OF_STEPS	SMALLINT	Die Anzahl von Schritten, die zur Verringerung der Datenabweichung auf null erforderlich sind

Tabelle 35. Felder der Funktion analyze_log_space (Forts.)

Spaltenname	Spaltentyp	Beschreibung
MAX_STEP_SIZE	BIGINT	Die maximale Größe von Daten (in Byte), die gleichzeitig versetzt werden können, ohne einen Fehler wegen eines gefüllten Protokolls zu verursachen

Tabelle 36. analyze_log_space - Eingabeparameter

Name	Datentyp	Beschreibung
inDBPGroup	VARCHAR(128)	Der Name der Datenbankpartitionsgruppe
inMainTbSchema	VARCHAR(128)	Das Schema der Haupttabelle
inMainTable	VARCHAR(128)	Die Haupttabelle in der Datenbankpartitionsgruppe (höchstwahrscheinlich die größte Tabelle in der Datenbankpartitionsgruppe)
useTbType	SMALLINT	Angabe für Analysetypen: SWRD_USE_STMG_TABLE 1: Gibt an, dass Informationen in Speicherverwaltungstabellen zur Ermittlung der Tabellenzeilenzahl pro Partitionen zu verwenden sind SWRD_USE_REALTIME_ANALYSIS 2: Gibt an, dass eine SELECT-Abfrage zur Ermittlung der Tabellenzeilenzahl pro Partition zu verwenden ist
inStmgTime	VARCHAR(26)	Die Zeitmarke für den Speicherverwaltungsdatensatz. Dieser Parameter wird ignoriert, wenn als Analysetyp der Wert SWRD_USE_REALTIME_ANALYSIS angegeben wird.
addDropOption	CHAR(1)	Angabe zum Hinzufügen oder Löschen von Partitionen 'A' = Partitionen hinzufügen 'D' = Partitionen löschen 'N' = Kein Hinzufügen oder Löschen
addDropList	VARCHAR(6000)	Hinzuzufügende oder zu löschende Partitionen, die in einer Zeichenfolge durch Kommas (,) getrennt aufgelistet werden
pNumber	VARCHAR(6000)	Eine Zeichenfolge, die alle Partitionsnummern enthält, die der Partitionsgewichtung entsprechen. Jede Partitionsnummer liegt zwischen 0 und 999, und die Nummern werden durch Kommas („,") getrennt. Leerzeichen sind in der Zeichenfolge nicht zulässig.
pWeight	VARCHAR(6000)	Eine Zeichenfolge, die alle Partitionsgewichtungen enthält, die der Benutzer entsprechend den Partitionsnummern in der Zeichenfolge pNumber angegeben hat. Jede Partitionsgewichtung ist eine Zahl zwischen 0 und SQL_MAXSMALLVAL, und die Zahlen werden durch Kommas („,") getrennt. Leerzeichen sind in der Zeichenfolge nicht zulässig.

Gespeicherte Prozedur generate_Distfile

Die Funktion generate_Distfile generiert eine Datenverteilungsdatei für die angegebene Tabelle und speichert sie unter dem angegebenen Dateinamen.

Tabelle 37. generate_Distfile - Eingabeparameter

Name	Datentyp	Beschreibung
inTbSchema	VARCHAR(128)	Der Schemaname der Tabelle
inTbName	VARCHAR(128)	Der Name der Tabelle

Tabelle 38. generate_Distfile - Ein- und Ausgabeparameter

Name	Datentyp	Beschreibung
fileName	VARCHAR(255)	Der Name der Datenverteilungsdatei. Wenn der angegebene Name nur ein Dateiname ist, wird die Datei im Verzeichnis exemplar/tmp gespeichert, und der vollständige Pfad- und Dateiname wird durch den Parameter zurückgegeben.

Gespeicherte Prozedur stepwise_redistribute_dbpg

Die Funktion stepwise_redistribute_dbpg führt die Umverteilung eines Teils der Datenbankpartitionsgruppe entsprechend der Eingabe und der Einstellungsdatei durch.

Schritt 1. Die Registrierdatenbank mit den Einstellungen wird über den inDbPGroup-Namen gesucht:

- Wenn die Registrierdatenbank nicht gefunden wird, wird ein Fehler zurückgegeben.
- Wenn die Registrierdatenbank gefunden wird, werden die folgenden Werte gelesen:
  - Der aktuelle Schritt
  - Die Anzahl von Schritten
  - Der Name der Partitionierungszuordnung
  - Die maximale Datengröße (die in jedem Schritt versetzt werden kann)
  - Der Processstatus
  - Die Partitionsgewichtungen

  Sie können die Daten der Datenbankpartitionsgruppe alle auf einmal umverteilen, indem Sie für inNumSteps den Wert -1 (SWRD_UNLIMITED_STEPS) angeben.

Schritt 2. Wenn der Prozessstatus in der Registrierdatenbank den Wert SWRD_STOP hat, wird der Prozess gestoppt und eine Warnung zurückgegeben. Wenn der Prozessstatus den Wert SWRD_CONTINUE hat, wird der Prozess fortgesetzt.

Schritt 3. Wenn inStartingPoint nicht NULL und gültig ist, wird der entsprechende Registrierungswert überschrieben. Anderenfalls wird der Registrierungswert für den aktuellen Schritt gelesen und als Startpunkt für diesen Schritt verwendet.

Schritt 4. Anschließend wird die Partitionierungszuordnung für den aktuellen Schritt generiert und die Partitionsgewichtungen werden aus der Registrierdatenbank mit den Einstellungen abgerufen. Wenn eine vorhandene *vollständige* Partitionierungszuordnung oder eine Verteilungsdatei angegeben wurde, wird die Partitionierungszuordnung entsprechend generiert.

Schritt 5. Das Umverteilungs-API wird mit der Option für die Partitionierungszuordnung aufgerufen. Wenn der Prozess abgeschlossen ist, wird der Registrierungswert für den nächsten Schritt erhöht.

Die Schritte 1 bis 5 werden für die Anzahl von Schritten wiederholt, die in der Registrierdatenbank angegeben ist.

Durch den Wert „-2" kann in dieser Prozedur angegeben werden, dass die Anzahl unbegrenzt ist.

Tabelle 39. stepwise_redistribute_dbpg - Eingabeparameter

Name	Datentyp	Beschreibung
inDBPGroup	VARCHAR(128)	Der Name der Zieldatenbankpartitionsgruppe
inStartingPoint	SMALLINT	Dieser Parameter kann den Wert NULL haben. Wenn er nicht null ist und eine positive Zahl angibt, wird er zum Überschreiben des Werts für "nextStep" verwendet, der durch die Registrierdatenbank der SWRD-Einstellungen angegeben wird. Dies kann eine nützliche Option sein, wenn Sie SWRD von einem bestimmten Schritt an erneut ausführen wollen.
inNumSteps	SMALLINT	Die Anzahl der auszuführenden Schritte. Wenn dieser Wert nicht null ist und eine positive Zahl angibt, wird er zum Überschreiben des Werts für "numSteps" verwendet, der durch die Registrierdatenbank der SWRD-Einstellungen angegeben wird. Dies kann eine nützliche Option sein, wenn Sie SWRD mit einer anderen Anzahl von Schritten ausführen wollen, als in den Einstellungen angegeben ist. Wenn zum Beispiel ein durch einen Zeitplan terminierter Arbeitsabschitt fünf Schritte hat, und der SWRD-Prozess bei Schritt 3 fehlgeschlagen ist, kann SWRD nach der Korrektur der Fehlerbedingung erneut aufgerufen werden, um die verbleibenden 3 Schritte auszuführen.

Benutzerdefinierte Funktion db_partitions

Die benutzerdefinierte Funktion db_partitions analysiert die Datei db2nodes.cfg und liefert eine Zeile für jede gefundene Partition zurück.

Eingabeparameter: keine

Tabelle 40. db_partitions - Ausgabeparameter

Name	Datentyp	Beschreibung
PARTITION_NUMBER	SMALLINT	Partitionsnummer
HOST_NAME	VARCHAR(128)	Hostname (für Intel-Plattformen ist dies der Maschinenname)
PORT_NUMBER	SMALLINT	Logische Portnummer
SWITCH_NAME	VARCHAR(128)	Netzswitchname (kann NULL sein)

Verwendungsbeispiel

Das folgende Beispiel zeigt eine Prozedur für den Befehlszeilenprozessor (CLP) unter AIX:

```
# ------------------------------------------------------------------------------------
# Angabe der Datenbank für die zu erstellende Verbindung
# ------------------------------------------------------------------------------------
dbName="SAMPLE"

# ------------------------------------------------------------------------------------
# Angabe des Namens für die Zieldatenbankpartitionsgruppe
# ------------------------------------------------------------------------------------
dbpgName="IBMDEFAULTGROUP"

# ------------------------------------------------------------------------------------
# Angabe des Schemas und des Namens der Tabelle
# ------------------------------------------------------------------------------------
tbSchema="$USER"
tbName="STAFF"

# ------------------------------------------------------------------------------------
# Angabe des Namens der Datenverteilungsdatei
# ------------------------------------------------------------------------------------
distFile="$HOME/sqllib/function/$dbName.IBMDEFAULTGROUP_swrdData.dst"

export DB2INSTANCE=$USER
export DB2COMM=TCPIP

# ------------------------------------------------------------------------------------
# Aufrufen der Aufrufanweisungen im Befehlszeilenprozessor (clp)
# ------------------------------------------------------------------------------------
db2start
db2 -v "connect to $dbName"

# ------------------------------------------------------------------------------------
# Analyse des Effekts einer hinzugefügten Partition ohne Anwendung der Änderungen  -
# Analyse eines hypothetischen Falls
#
# - Im folgenden Fall werden hypothetisch die Partitionen 40, 50 und 60 der
#   Datenbankpartitionsgruppe hinzugefügt; für die Partitionen 10,20,30,40,50,60 werden
#   die Zielverhältnisse 1:2:1:2:1:2 verwendet.
#
# Hinweis: In diesem Beispiel sind nur die Partitionen 10, 20 und 30 tatsächlich in der
#          Datenbankpartitionsgruppe vorhanden.
# ------------------------------------------------------------------------------------
db2 -v "call sysproc.analyze_log_space('$dbpgName', '$tbSchema', '$tbName', 2, ' ',
'A', '40,50,60', '10,20,30,40,50,60', '1,2,1,2,1,2')"
```

```
# ------------------------------------------------------------------------------
# Analyse des Effekts einer gelöschten Partition ohne Anwendung der Änderungen
#
# - Im folgenden Fall wird hypothetisch die Partition 30 aus der Datenbankpartitionsgruppe
#   gelöscht und die Daten werden auf die Partitionen 10 und 20 unter Verwendung
#   eines Zielverhältnisses von 1 : 1 umverteilt.
#
# Hinweis: In diesem Beispiel sollten alle Partitionen, 10, 20 und 30, in der
#          Datenbankpartitionsgruppe vorhanden sein.
# ------------------------------------------------------------------------------
db2 -v "call sysproc.analyze_log_space('$dbpgName', '$tbSchema', '$tbName', 2, ' ',
'D', '30', '10,20','1,1')"

# ------------------------------------------------------------------------------
# Generieren einer Datenverteilungsdatei zur Verwendung durch den Umverteilungsprozess
# ------------------------------------------------------------------------------
db2 -v "call sysproc.generate_distfile('$tbSchema', '$tbName', '$distFile')"

# ------------------------------------------------------------------------------
# Schreiben eines schrittweisen Umverteilungsplan in eine Registrierdatenbank
#
# Der Wert 1 für den zehnten Parameter kann bewirken, dass die zurzeit ausgeführte
# gespeicherte Prozedur zur schrittweisen Umverteilung den aktuellen Schritt beendet und
# die Verarbeitung stoppt, bis dieser Parameter auf 0 zurückgesetzt wird und die
# gespeicherte Prozedur zur Umverteilung erneut aufgerufen wird.
# ------------------------------------------------------------------------------
db2 -v "call sysproc.set_swrd_settings('$dbpgName', 255, 0, ' ', '$distFile', 1000,
12, 2, 1, 0, '10,20,30', '50,50,50')"

# ------------------------------------------------------------------------------
# Berichten des Inhalts des schrittweisen Umverteilungsplans für die angegebene
# Datenbankpartitionsgruppe
# ------------------------------------------------------------------------------
db2 -v "call sysproc.get_swrd_settings('$dbpgName', 255, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)"

# ------------------------------------------------------------------------------
# Umverteilen der Daten der Datenbankpartitionsgruppe "dbpgName" entsprechend dem in
# der Registrierdatenbank durch set_swrd_settings gespeicherten Umverteilungsplan.
# Mit Start bei Schritt 3 werden die Daten umverteilt, bis zwei Schritte im
# Umverteilungsplan ausgeführt sind.
# ------------------------------------------------------------------------------
db2 -v "call sysproc.stepwise_redistribute_dbpg('$dbpgName', 3, 2)"
```

Zugehörige Konzepte:

- „Umverteilung von Daten” auf Seite 341

Kapitel 12. Durchführen von Vergleichstests

Dieses Kapitel erläutert den Vergleichstestprozess sowie die Verwendung des Dienstprogramms *db2batch* zur Durchführung eines Vergleichstests mit einer Datenbankauslastung.

Durchführen von Vergleichstests

Die Durchführung von Vergleichstests (Benchmarktests) ist ein natürlicher Bestandteil des Entwicklungszyklus für Anwendungen. Sie erfordert die Zusammenarbeit von Anwendungsentwicklern und Datenbankadministratoren (DBAs) und sollte für eine Anwendung stattfinden, um Daten über die aktuelle Leistung zu erhalten und Ansätze zur Leistungsoptimierung zu ermitteln. Wenn der Code einer Anwendung bereits mit größtmöglicher Effizienz arbeitet, können eventuell weitere Leistungsvorteile durch die Optimierung der Konfigurationsparameter der Datenbank und des Datenbankmanagers realisiert werden. Sie können außerdem Anwendungsparameter optimieren, um die Anforderungen einer Anwendung besser zu erfüllen.

Durch verschiedene Arten von Vergleichstests lassen sich bestimmte Arten von Informationen zu gewinnen:

- Ein Vergleichstest der Art *Transaktion pro Sekunde* liefert Anhaltspunkte über die Leistungskapazität des Datenbankmanagers unter bestimmten, eingeschränkten Laborbedingungen.
- Ein *Anwendungsvergleichstest* prüft die gleichen Durchsatzkapazitäten unter Bedingungen, die den Produktionsbedingungen näher kommen.

Die Optimierung von Konfigurationsparametern durch Vergleichstests legt diese „Realbedingungen" zugrunde und erfordert ein wiederholtes Ausführen von SQL aus der Anwendung mit variierenden Parameterwerten, bis die Anwendung mit der höchstmöglichen Effizienz arbeitet.

Die hier beschriebenen Vergleichstestmethoden sind auf die Optimierung von Konfigurationsparametern ausgerichtet. Jedoch kann derselbe Grundansatz auch für die Optimierung anderer, die Leistung beeinflussender Faktoren herangezogen werden, wie zum Beispiel:

- SQL-Anweisungen
- Indizes
- Tabellenbereichskonfiguration
- Anwendungscode
- Hardwarekonfiguration

Vergleichstests geben Aufschluss darüber, wie der Datenbankmanager unter unterschiedlichen Bedingungen reagiert. Es können Szenarios entwickelt werden, um die Behandlung gegenseitiger Sperren, die Leistung von Dienstprogrammen, die verschiedenen Methoden zum Laden von Daten, die Transaktionsgeschwindigkeit bei wachsenden Benutzerzahlen und auch die Auswirkungen der Verwendung eines neuen Produktrelease auf die Anwendung zu testen.

Vergleichstestmethoden

Bei der Durchführung von Vergleichstests wird eine reproduzierbare Umgebung zugrunde gelegt, so dass der gleiche Test unter den gleichen Bedingungen Ergebnisse liefert, deren Vergleich legitim ist.

Sie können mit den Vergleichstests beginnen, indem Sie die Testanwendung in einer Normalumgebung ausführen. Wenn Sie ein Leistungsproblem orten, können Sie spezielle Testszenarios entwickeln, die den Wirkungsbereich der getesteten Funktion begrenzen. Die speziellen Testszenarios brauchen nicht eine gesamte Anwendung zu emulieren, um wertvolle Informationen zu liefern. Es empfiehlt sich, mit einfachen Messungen zu beginnen und die Komplexität der Methoden nur dann zu erhöhen, wenn dies erforderlich ist.

Gute Vergleichstests oder Messungen besitzen folgende Merkmale:
- Die Tests sind wiederholbar.
- Jede Wiederholung eines Tests beginnt im gleichen Systemstatus.
- Es sind keine anderen Funktionen bzw. Anwendungen im System aktiv, sofern vom Testszenario nicht vorgesehen ist, dass bestimmte andere Systemaktivitäten gleichzeitig stattfinden.

  Anmerkung: Gestartete Anwendungen belegen Speicher, selbst wenn Sie auf Symbolgröße verkleinert wurden oder momentan inaktiv sind. Dadurch erhöht sich die Wahrscheinlichkeit, dass Seitenauslagerungen zu einer Verzerrung der Vergleichstestergebnisse führen, wodurch die Wiederholbarkeitsregel verletzt wird.
- Die Hardware und die Software, die für die Vergleichstests verwendet werden, entsprechen der realen Produktionsumgebung.

Zur Durchführung von Vergleichstests erstellen Sie ein Szenario und führen die Anwendungen mehrere Male in diesem Szenario aus, indem Sie bei jeder Ausführung wichtige Informationen erfassen. Die Erfassung wichtiger Informationen nach jeder Ausführung ist von höchster Bedeutung bei der Ermittlung der Änderungen, die eine Leistungsverbesserung sowohl für die Anwendung als auch für die Datenbank bewirken können.

Zugehörige Konzepte:

- „Vorbereiten von Vergleichstests" auf Seite 358
- „Erstellung von Vergleichstests" auf Seite 360
- „Ausführung von Vergleichstests" auf Seite 366
- „Vergleichstestanalyse - Beispiel" auf Seite 368

Vorbereiten von Vergleichstests

Schließen Sie den logischen Entwurf der Datenbank ab, für die Sie die Anwendung ausführen, bevor Sie mit den Leistungsvergleichstests beginnen. Richten Sie Tabellen, Sichten und Indizes ein und füllen Sie sie mit Daten. Normalisieren Sie Tabellen, binden Sie Anwendungspakete und füllen Sie Tabellen mit realistischen Daten. Das endgültige physische Modell der Datenbank sollte bereits ebenfalls feststehen. Platzieren Sie Datenbankmanagerobjekte an ihre endgültigen Datenträgerpositionen, definieren Sie die Größe von Protokolldateien, legen Sie die Position von Arbeitsdateien und Sicherungen fest und testen Sie die Sicherungsprozeduren. Prüfen Sie ferner Ihre Pakete, um sicherzustellen, dass Leistungsoptionen wie Zeilenblockung aktiviert werden, wenn dies möglich ist.

Sie sollten einen Stand der Programmier- und Testphasen für die Anwendung erreicht haben, der es Ihnen ermöglicht, die Vergleichstestprogramme zu erstellen. Während der Vergleichstests können die praktischen Grenzen einer Anwendungen zutage treten. Jedoch liegt das Ziel der hier beschriebenen Vergleichstests in der Messung der Leistung und nicht in der Feststellung von Fehlern oder abnormalen Beendigungsbedingungen.

Das Vergleichstestprogramm muss in einer möglichst wirklichkeitsnahen Nachbildung der tatsächlichen Produktionsumgebung ausgeführt werden. Im Idealfall sollte ein Server des gleichen Modells mit derselben Speicher- und Festplattenkonfiguration verwendet werden. Dies ist besonders dann von Bedeutung, wenn die Anwendung letztendlich eine große Anzahl von Benutzern und große Mengen von Daten verarbeiten soll. Das Betriebssystem und mögliche Einrichtungen der Datenübertragung oder des Dateiservice sollten ebenfalls bereits optimiert worden sein.

Stellen Sie sicher, dass Sie Vergleichstests mit einer Datenbank in Größe der Produktionsdatenbank durchführen. Eine einzelne SQL-Anweisung sollte die gleiche Menge an Daten liefern und den gleichen Aufwand für Sortiervorgänge erfordern wie in der Geschäftsumgebung. Diese Regel gewährleistet, dass die Anwendung repräsentative Speicheranforderungen testet.

Die zu testenden SQL-Anweisungen sollten entweder zur Kategorie *repräsentatives SQL* oder zur Kategorie *Extremfall-SQL* (Worst-Case) gehören, wie im Folgenden erläutert wird:

Repräsentatives SQL

Zu repräsentativem SQL werden solche Anweisungen gezählt, die während eines typischen Einsatzes der zu testenden Anwendung ausgeführt werden. Die Anweisungen, die zum Test ausgewählt werden, sind von der Art der Anwendung abhängig. Beispielsweise kann für eine Dateneingabeanwendung eine Anweisung INSERT getestet werden, während für eine Banktransaktion eine Anweisung FETCH, eine Anweisung UPDATE und mehrere Anweisungen INSERT getestet werden können. Die Häufigkeit, mit der die Anwendung ausgeführt wird, und der Umfang der von den ausgewählten Anweisungen verarbeiteten Daten sollte als durchschnittlich angesehen werden können. Wenn die aufgrund der Anweisungen zu verarbeitenden Datenmengen sehr umfangreich sind, sind diese Anweisungen unter der Kategorie *Extremfall-SQL* zu betrachten, selbst wenn es sich um typische SQL-Anweisungen handelt.

Extremfall-SQL

Zu dieser Kategorie gehören Anweisungen mit folgenden Merkmalen:

- Anweisungen, die häufig ausgeführt werden
- Anweisungen, durch die umfangreiche Datenmengen verarbeitet werden
- Anweisungen, die zeitkritisch sind

  Ein Beispiel hierfür ist eine Anwendung, die ausgeführt wird, wenn ein Telefonanruf von einem Kunden eingeht, und deren Anweisungen ausgeführt werden müssen, um Daten des Kunden abzurufen oder zu aktualisieren, während der Kunde wartet.
- Anweisungen mit der größten Anzahl zu verknüpfender Tabellen, oder mit dem komplexesten SQL in der Anwendung

  Ein Beispiel wäre eine Bankanwendung, die kombinierte Kontoauszüge über die monatlichen Kontobewegungen für sämtliche verschiedene Arten von Konten eines Kunden erstellt. Eine allgemeine Tabelle könnte vielleicht die Kundenadressen und die Kontonummern enthalten; jedoch

müssen mehrere andere Tabellen verknüpft werden, um alle benötigten Daten über Kontotransaktionen verarbeiten und zusammenstellen zu können. Die Multiplikation des Aufwands, der für ein Konto erforderlich ist, mit den mehreren Tausend Konten, die im gleichen Zeitraum verarbeitet werden müssen, macht deutlich, das jede potenzielle Zeiteinsparung die Leistungsanforderungen erhöht.

- Anweisungen, die einen ungünstigen Zugriffspfad verwenden, z. B. eine Anweisung, die nicht sehr oft ausgeführt wird und nicht von den Indizes unterstützt wird, die für die betroffenen Tabellen erstellt wurden
- Anweisungen, die über einen langen Zeitraum hinweg laufen
- Eine Anweisung, die nur bei der Initialisierung einer Anwendung ausgeführt wird, jedoch überproportionale Ressourcenanforderungen stellt

  Ein Beispiel wäre eine Anwendung, die eine Liste der Arbeiten für Konten erstellt, die während des Arbeitstages auszuführen sind. Wenn die Anwendung gestartet wird, löst die erste größere SQL-Anweisung eine Verknüpfung über zahlreiche Tabellen aus, um eine sehr umfangreiche Liste aller Konten zu erstellen, für die der Benutzer der Anwendung verantwortlich ist. Die Anweisung wird vielleicht nur wenige Male am Tag ausgeführt, aber ihre Ausführung nimmt einige Minuten in Anspruch, wenn sie nicht ausreichend optimiert wurde.

Zugehörige Konzepte:

- „Durchführen von Vergleichstests" auf Seite 357
- „Erstellung von Vergleichstests" auf Seite 360

Erstellung von Vergleichstests

Beim Entwurf und bei der Implementierung eines Vergleichstestprogramms sind eine Reihe von Faktoren zu beachten. Da der Hauptzweck des Programms darin besteht, eine Benutzeranwendung zu simulieren, ist die allgemeine Struktur des Programms unterschiedlich. Sie können die gesamte Anwendung zum Vergleichstest verwenden, so dass Sie nur die entsprechenden Mittel zur Erfassung der Zeiten für die zu analysierenden SQL-Anweisungen einfügen. Bei großen oder komplexen Anwendungen ist es eventuell praktischer, nur die Blöcke mit den wichtigen Anweisungen in das Vergleichstestprogramm aufzunehmen.

Zum Testen der Leistung bestimmter SQL-Anweisungen können Sie diese Anweisungen allein in das Vergleichstestprogramm aufnehmen und die benötigten Anweisungen CONNECT, PREPARE, OPEN und andere sowie Mechanismen zur Zeiterfassung hinzufügen.

Ein weiterer wichtiger Gesichtspunkt ist die Art des Vergleichs, die verwendet werden sollte. Eine Möglichkeit ist die, eine Gruppe von SQL-Anweisungen über ein Zeitintervall wiederholt auszuführen. Das Verhältnis der Anzahl der ausgeführten Anweisungen zu diesem Zeitintervall ergäbe einen Wert für den Durchsatz für die Anwendung. Eine andere Möglichkeit ist die, einfach die für die Ausführung einzelner SQL-Anweisungen erforderliche Zeit zu bestimmen.

Für alle Vergleichstests benötigen Sie ein effizientes Zeiterfassungssystem, um die während der Verarbeitung entweder einzelner SQL-Anweisungen oder der gesamten Anwendung abgelaufene Zeit zu berechnen. Bei der Simulation von Anwendungen, in denen einzelne SQL-Anweisungen isoliert ausgeführt werden, kann es wichtig sein, die Zeiten für die Anweisungen CONNECT, PREPARE und COMMIT zu verfolgen. Bei Programmen jedoch, die viele verschiedene Anweisungen verar-

beiten, wird vielleicht nur eine einzige Anweisung CONNECT oder COMMIT benötigt, so dass die Erfassung der Ausführungszeit für eine einzelne Anweisung Priorität haben könnte.

Obwohl die Erfassung der benötigten Zeit für jede Abfrage einen wichtigen Faktor bei der Leistungsanalyse darstellt, werden durch sie nicht unbedingt potenzielle Leistungsengpässe offen gelegt. Zum Beispiel könnten Informationen über die CPU-Auslastung, über aktive Sperren und Pufferpoolein-/ausgaben Hinweise darauf geben, dass eine Anweisung durch die Ein-/Ausgabeaktivitäten gebremst wird und nicht die volle Kapazität der CPU nutzt. Ein Vergleichstestprogramm sollte es ermöglichen, diese Art von Daten für eine detailliertere Analyse bei Bedarf zu erfassen.

Nicht alle Anwendungen senden die gesamte Menge der durch eine Abfrage abgerufenen Zeilen an eine Ausgabeeinheit. Zum Beispiel könnte die gesamte Antwortmenge als Eingabe für ein anderes Programm dienen, so dass keine Zeilen aus der ersten Anwendung als Ausgabe gesendet werden. Die Formatierung von Daten zur Ausgabe auf der Anzeige verursacht in der Regel einen hohen CPU-Aufwand und spiegelt den Benutzerbedarf nicht unbedingt wider. Um eine genaue Simulation zu erhalten, sollte ein Vergleichstestprogramm die Zeilenbehandlung der bestimmten Anwendung berücksichtigen. Wenn Zeilen an eine Ausgabeeinheit gesendet werden, könnte ineffizientes Formatieren den Hauptanteil der CPU-Verarbeitungszeit in Anspruch nehmen und so zu einer Verzerrung der tatsächlichen Leistungsdaten für die SQL-Anweisung als solche führen.

Das Vergleichstest-Tool db2batch: Es steht ein Vergleichstest-Tool (`db2batch`) im Unterverzeichnis `bin` des Verzeichnisses `sqllib` Ihres Exemplars zur Verfügung. Dieses Tool setzt viele der Richtlinien zur Erstellung eines Vergleichstestprogramms um. Dieses Tool kann SQL-Anweisungen entweder aus einer unstrukturierten Datei oder von der Standardeingabeeinheit lesen, die Anweisungen dynamisch beschreiben und vorbereiten und eine Antwortmenge zurückliefern. Es gibt Ihnen außerdem die Möglichkeit, die Größe der Antwortmenge und die Anzahl der Zeilen, die aus dieser Anwortmenge an eine Ausgabeeinheit gesendet werden, zu steuern.

Sie können die Stufe der leistungsbezogenen Daten angeben, die geliefert werden sollen, einschließlich der benötigten Zeit, CPU- und Pufferpoolauslastung, Sperren sowie anderer statistischer Daten aus dem Datenbankmonitor. Bei der Zeitmessung für eine Gruppe von SQL-Anweisungen erstellt `db2batch` auch eine Ergebnisübersicht und berechnet arithmetische und geometrische Mittelwerte. Sie können Syntaxoptionen anzeigen, indem Sie den Befehl `db2batch -h` in eine Befehlszeile eingeben.

Dieses Vergleichstest-Tool verfügt darüber hinaus über die Option CLI. Mit dieser Option können Sie die Größe eines Cache angeben. Im folgenden Beispiel wird `db2batch` im CLI-Modus mit einer Cachegröße von 30 Anweisungen ausgeführt:

```
db2batch -d sample -f db2batch.sql -cli 30
```

Auch die Fernausführung von db2batch ist möglich. Wenn Sie entweder den Befehlsparameter

```
-f <dateiname>
```

oder

```
-o <optionen>
```

des Vergleichstest-Tools verwenden, gilt Folgendes:

- Die Steuerungsoptionen

  `perf_detail`

  und

  `-p <perf_detail>`

  (die die Stufe der zuzurückzugebenden Leistungsdaten angeben) werden bei der Fernausführung nicht unterstützt, wenn sie auf einen Wert größer 1 gesetzt werden.

Bei anderen als diesen beiden Optionen werden die Steuerungsoptionswerte

`perf_detail`

und

`-p <perf_detail>`

unterstützt und sind für alle DB2® Universal Database-Plattformen gültig.

Beispiele für db2batch-Tests

Das folgende Beispiel zeigt, wie das Tool `db2batch` mit einer Eingabedatei `db2batch.sql` verwendet werden könnte:

```
-- db2batch.sql
-- ------------
--#SET PERF_DETAIL 3 ROWS_OUT 5

-- This query lists employees, the name of their department
-- and the number of activities to which they are assigned for
-- employees who are assigned to more than one activity less than
-- full-time.
--#COMMENT Query 1
select lastname, firstnme,
deptname, count(*) as num_act
from employee, department, emp_act
where employee.workdept = department.deptno and
employee.empno = emp_act.empno and
emp_act.emptime < 1
group by lastname, firstnme, deptname
having count(*) > 2;
--#SET PERF_DETAIL 1 ROWS_OUT 5
--#COMMENT Query 2
select lastname, firstnme,
deptname, count(*) as num_act
from employee, department, emp_act
where employee.workdept = department.deptno and
employee.empno = emp_act.empno and
emp_act.emptime < 1
group by lastname, firstnme, deptname
having count(*) <= 2;
```

Abbildung 26. Beispieleingabedatei für das Vergleichstest-Tool: db2batch.sql

Geben Sie zum Beispiel den folgenden Aufruf für das Vergleichstest-Tool ein:

```
db2batch -d sample -f db2batch.sql
```

Dadurch wird die folgende Ausgabe erstellt:

```
--#SET PERF_DETAIL 3 ROWS_OUT 5
Query 1

Statement number: 1

select lastname, firstnme,
deptname, count(*) as num_act
from employee, department, emp_act
where employee.workdept = department.deptno and
employee.empno = emp_act.empno and
emp_act.emptime < 1
group by lastname, firstnme, deptname
having count(*) > 2
```

Abbildung 27. Beispielausgabe des Programms db2batch (Teil 1)

```
LASTNAME         FIRSTNME      DEPTNAME                       NUM_ACT
-------------------------------------------------------------------------
JEFFERSON        JAMES         ADMINISTRATION SYSTEMS                3
JOHNSON          SYBIL         ADMINISTRATION SYSTEMS                4
NICHOLLS         HEATHER       INFORMATION CENTER                    4
PEREZ            MARIA         ADMINISTRATION SYSTEMS                4
SMITH            DANIEL        ADMINISTRATION SYSTEMS                7
Number of rows retrieved is:       5
Number of rows sent to output is:  5
Elapsed Time is:           0.074      seconds
Locks held currently                        = 0
Lock escalations                            = 0
Total sorts                                 = 5
Total sort time (ms)                        = 0
Sort overflows                              = 0
Buffer pool data logical reads              = 13
Buffer pool data physical reads             = 5
Buffer pool data writes                     = 0
Buffer pool index logical reads             = 3
Buffer pool index physical reads            = 0
Buffer pool index writes                    = 0
Total buffer pool read time (ms)            = 23
Total buffer pool write time (ms)           = 0
Asynchronous pool data page reads           = 0
Asynchronous pool data page writes          = 0
Asynchronous pool index page reads          = 0
Asynchronous pool index page writes         = 0
Total elapsed asynchronous read time        = 0
Total elapsed asynchronous write time       = 0
Asynchronous read requests                  = 0
LSN Gap cleaner triggers                    = 0
Dirty page steal cleaner triggers           = 0
Dirty page threshold cleaner triggers       = 0
Direct reads                                = 8
Direct writes                               = 0
Direct read requests                        = 4
Direct write requests                       = 0
Direct read elapsed time (ms)               = 0
Direct write elapsed time (ms)              = 0
Rows selected                               = 5
Log pages read                              = 0
Log pages written                           = 0
Catalog cache lookups                       = 3
Catalog cache inserts                       = 3
Buffer pool data pages copied to ext storage    = 0
Buffer pool index pages copied to ext storage   = 0
Buffer pool data pages copied from ext storage  = 0
Buffer pool index pages copied from ext storage = 0
Total Agent CPU Time (seconds)              = 0.02
Post threshold sorts                        = 0
Piped sorts requested                       = 5
Piped sorts accepted                        = 5
```

Abbildung 28. Beispielausgabe des Programms db2batch (Teil 1)

```
--#SET PERF_DETAIL 1 ROWS_OUT 5
Query 2
Statement number: 2
select lastname, firstnme,
deptname, count(*) as num_act
from employee, department, emp_act
where employee.workdept = department.deptno and
employee.empno = emp_act.empno and
emp_act.emptime < 1
group by lastname, firstnme, deptname
having count(*) <= 2
LASTNAME        FIRSTNME     DEPTNAME                       NUM_ACT
------------------------------------------------------------------------
GEYER           JOHN         SUPPORT SERVICES                     2
GOUNOT          JASON        SOFTWARE SUPPORT                     2
HAAS            CHRISTINE    SPIFFY COMPUTER SERVICE DIV.         2
JONES           WILLIAM      MANUFACTURING SYSTEMS                2
KWAN            SALLY        INFORMATION CENTER                   2
Number of rows retrieved is:       8
Number of rows sent to output is:  5
Elapsed Time is:           0.037      seconds
Summary of Results
==================
                Elapsed             Agent CPU         Rows      Rows
Statement #     Time (s)            Time (s)          Fetched   Printed
1                    0.074               0.020          5         5
2                    0.037       Not Collected          8         5
Arith. mean     0.055
Geom.  mean     0.052
```

Abbildung 29. Beispielausgabe des Programms db2batch (Teil 2)

Die gezeigte Beispielausgabe umfasst spezifische Datenelemente, die vom Datenbanksystemmonitor geliefert werden.

Im folgenden Beispiel (unter UNIX®) wird lediglich die Ergebnisübersicht (Summary of Results) erstellt.

```
db2batch -d sample -f db2batch.sql -r /dev/null,
```

Mit diesem Aufruf wird nur die Ergebnisübersicht angezeigt. Durch die Option `-r` wird die Ausgabedatei 1 (`outfile1`) durch `/dev/null` ersetzt und die Ausgabedatei 2 (`outfile2`), die nur die Ergebnistabelle enthält, bleibt leer, so dass die Ausgabe von `db2batch` an die Anzeige gesendet wird:

```
Summary of Results
==================
                Elapsed             Agent CPU         Rows      Rows
Statement #     Time (s)            Time (s)          Fetched   Printed
1                    0.074               0.020          5         5
2                    0.037       Not Collected          8         5
Arith. mean     0.055
Geom.  mean     0.052
```

Abbildung 30. Beispielausgabe des Programms db2batch - nur Ergebnistabelle

Zugehörige Konzepte:

- „Erstellung von Vergleichstests" auf Seite 360
- „Ausführung von Vergleichstests" auf Seite 366
- „Vergleichstestanalyse - Beispiel" auf Seite 368

Ausführung von Vergleichstests

Für eine Art von Datenbankvergleichstest wählen Sie einen Konfigurationsparameter aus und führen den Test mit verschiedenen Werten für den gewählten Parameter aus, bis die maximale Leistungssteigerung erzielt ist. Ein einzelner Test sollte die mehrmalige Ausführung der Anwendung (z. B. zwanzig- oder dreißigmal) mit demselben Parameterwert beinhalten, um einen Durchschnittswert für die benötigte Zeit zu ermitteln, der die Auswirkungen einer Änderung des Parameterwertes klarer wiedergibt.

Bei der Ausführung des Vergleichstests sollte der erste Durchlauf, der so genannte Aufwärmdurchlauf (Warm-up Run) von den nachfolgenden Iterationen, d. h. den Normaldurchläufen (Normal Run), getrennt betrachtet werden. Da der Aufwärmdurchlauf einige Startaktivitäten, wie zum Beispiel die Initialisierung des Pufferpools, durchführt, dauert er etwas länger als Normaldurchläufe. Obwohl die Daten aus dem Aufwärmdurchlauf durchaus realistische Werte darstellen können, sind sie für die statistische Auswertung nicht relevant. Zur Berechnung des Durchschnittswerts für die benötigte Zeit oder die CPU-Auslastung für eine bestimmte Gruppe von Parameterwerten sollten Sie nur die Ergebnisse aus Normaldurchläufen verwenden.

Sie können auch in Betracht ziehen, mit dem *Konfigurationsadvisor* den Aufwärmdurchlauf des Vergleichstests zu erstellen. Die im *Konfigurationsadvisor* gestellten Fragen geben einen Einblick in die Gesichtspunkte, die bei der Anpassung der Konfiguration Ihrer Umgebung für Normaldurchläufe in der Vergleichstestphase zu beachten sind. Sie können den *Konfigurationsadvisor* über die Steuerzentrale oder durch Ausführen des Befehls `db2 autoconfigure` mit den entsprechenden Optionen starten.

Wenn Vergleichstests mit einzelnen Abfragen ausgeführt werden, müssen Sie sicherstellen, dass Sie die potenziellen Auswirkungen früherer Abfragen minimieren, indem Sie den Pufferpool reinigen. Zum Reinigen des Pufferpools füllen Sie ihn mit einer Anzahl von Seiten, die für die Abfrage irrelevant sind (Flushing).

Nach der Ausführung der Durchläufe für eine bestimmte Gruppe von Parameterwerten können Sie den Wert eines einzelnen Parameters ändern. Zwischen den einzelnen Durchläufen müssen Sie jedoch folgende Maßnahmen durchführen, um die Vergleichstestumgebung wieder in den Ausgangszustand zurückzusetzen:

- . Wenn die Katalogstatistiken für den Test aktualisiert wurden, stellen Sie sicher, dass für jeden Durchlauf dieselben Werte für die Statistiken verwendet werden.
- Die Daten, die im Test verwendet werden, müssen konsistent sein, wenn sie durch die Tests aktualisiert werden. Dies kann folgendermaßen sichergestellt werden:
  - Durch Verwenden des Dienstprogramms RESTORE, um die gesamte Datenbank wieder herzustellen. Die Sicherungskopie der Datenbank enthielte den früheren Zustand, der für den nächsten Test bereit ist.
  - Durch Verwenden des Dienstprogramms IMPORT oder LOAD, um eine exportierte Kopie der Daten wiederherzustellen. Diese Methode erlaubt die Wiederherstellung nur der Daten, die vom Test betroffen waren. Die Dienstprogramme REORG und RUNSTATS sollten für die Tabellen und Indizes, die diese Daten enthalten, ausgeführt werden.
- Binden Sie die Anwendung erneut an die Datenbank, um sie in ihren Ausgangszustand zurückzusetzen.

Zusammenfassung: Führen Sie die folgenden Schritte oder Durchläufe aus, um Vergleichstests für eine Datenbankanwendung durchzuführen:

Schritt 1

Belassen Sie alle Parameter zur Optimierung der Datenbank und des Datenbankmanagers außer folgenden auf ihren **Standardwerten**:

- Die Parameter, die für die Belastung durch den Test und die Zielsetzung des Tests von Bedeutung sind. (Sie werden selten genügend Zeit haben, um Vergleichstests zur Optimierung aller Parameter durchzuführen, so dass es empfehlenswert ist, zu Beginn gute Schätzwerte für einige Parameter festzulegen und von diesen ausgehend die Optimierung vorzunehmen.)
- Protokolldateigrößen, die während der Einheiten- oder Systemtests für Ihre Anwendung festgelegt werden sollten.
- Alle die Parameter, die geändert werden müssen, damit die Anwendung ausgeführt werden kann (d. h., Änderungen zur Verhinderung negativer SQL-Rückkehrcodes aufgrund von Ereignissen wie Speicherknappheit für den Anweisungszwischenspeicher).

Führen Sie Ihre Anzahl von Durchläufen (Iterationen) für diesen Anfangszustand aus, und berechnen Sie den Durchschnittswert für die Zeitmesswerte oder die CPU.

Schritt 2

Wählen Sie einen und nur einen für die Optimierung zu testenden Parameter aus, und ändern Sie seinen Wert.

Schritt 3

Führen Sie eine weitere Anzahl von Durchläufen (Iterationen) aus, und berechnen Sie den Durchschnittswert für die Zeitmesswerte oder die CPU.

Schritt 4

Ergreifen Sie in Abhängigkeit von den Ergebnissen des Vergleichstests eine der folgenden Maßnahmen:

- Wenn die Leistung besser wird, ändern Sie den Wert desselben Parameters und kehren zu Schritt 3 zurück. Ändern Sie diesen Parameter so lange, bis der maximale Leistungswert gezeigt wird.
- Wenn die Leistung sinkt oder unverändert bleibt, setzen Sie den Parameter auf seinen vorigen Wert zurück, kehren zu Schritt 2 zurück und wählen einen anderen Parameter. Wiederholen Sie diese Prozedur, bis alle Parameter getestet sind.

  Anmerkung: Wenn Sie die Leistungsergebnisse in grafischer Darstellung festhalten würden, müssten Sie nach Stellen suchen, an denen die Kurve einen Maximalwert erreicht und dort verbleibt bzw. wieder absinkt.

Sie können ein Treiberprogramm schreiben, das Sie bei der Durchführung der Vergleichstests unterstützt. Dieses Treiberprogramm könnte in einer Sprache wie REXX bzw. bei auf UNIX® basierenden Systemen mit Hilfe von Shellprozeduren (Scripts) erstellt werden.

Dieses Treiberprogramm könnte das Vergleichstestprogramm ausführen, ihm dabei die richtigen Parameter übergeben, den Test durch mehrere Durchläufe führen, die Umgebung in einen konsistenten Zustand zurückversetzen, den nächsten Test mit neuen Parameterwerten vorbereiten und die Testdaten sammeln bzw. konsolidieren.

Diese Treiberprogramme können so flexibel gestaltet werden, dass sie zur Ausführung einer ganzen Reihe von Vergleichstests, zur Analyse der Ergebnisse und zur Erstellung eines Berichts über die endgültigen und optimalen Parameterwerte für einen bestimmten Test verwendet werden könnten.

Zugehörige Konzepte:

- „Durchführen von Vergleichstests" auf Seite 357
- „Vorbereiten von Vergleichstests" auf Seite 358
- „Beispiele für db2batch-Tests" auf Seite 362

Vergleichstestanalyse - Beispiel

Die Ausgabe des Vergleichstestprogramms sollte eine Kennung für jeden Test (Test Number), die Iteration (Durchlauf) der Programmausführung (Iteration Number), die Anweisungsnummer (Statement Number) und die für die Ausführung erfasste Zeit (Timing) enthalten.

Eine Übersicht über Vergleichstestergebnisse nach einer Reihe von Messungen könnte wie folgt aussehen:

```
Test     Iter.   Stmt    Timing         SQL Statement
Numbr    Numbr   Numbr   (hh:mm:ss.ss)
 002      05      01     00:00:01.34    CONNECT TO SAMPLE
 002      05      10     00:02:08.15    OPEN cursor_01
 002      05      15     00:00:00.24    FETCH cursor_01
 002      05      15     00:00:00.23    FETCH cursor_01
 002      05      15     00:00:00.28    FETCH cursor_01
 002      05      15     00:00:00.21    FETCH cursor_01
 002      05      15     00:00:00.20    FETCH cursor_01
 002      05      15     00:00:00.22    FETCH cursor_01
 002      05      15     00:00:00.22    FETCH cursor_01
 002      05      20     00:00:00.84    CLOSE cursor_01
 002      05      99     00:00:00.03    CONNECT RESET
```

Abbildung 31. Beispielergebnisse eines Vergleichstestprogramms

Anmerkung: Die Daten im gezeigten Bericht dienen nur der Illustration. Sie stellen **keine** durch Tests ermittelten Ergebnisse dar.

Eine Analyse zeigt, dass die Anweisung CONNECT (Anweisung 01) 1,34 Sekunden dauerte, die Anweisung OPEN CURSOR (Anweisung 10) 2 Minuten und 8,15 Sekunden, die Anweisungen FETCH (Anweisung 15) sieben Zeilen mit der längsten Verzögerung von 0,28 Sekunden lieferten, die Anweisung CLOSE CURSOR (Anweisung 20) 0,84 Sekunden benötigte und die Anweisung CONNECT RESET (Anweisung 99) 0,03 Sekunden in Anspruch nahm.

Wenn das Programm die Daten in einem DEL-Format (Delimited ASCII) ausgeben könnte, könnten diese später in eine Datenbanktabelle oder ein Arbeitsblatt zur weiteren statistischen Analyse importiert werden.

Eine Beispielausgabe für einen Vergleichstestbericht könnte folgendermaßen aussehen:

```
PARAMETER        VALUES FOR EACH BENCHMARK TEST
TEST NUMBER      001     002     003    004     005
locklist         63      63      63      63      63
maxappls         8       8       8       8       8
applheapsz       48      48      48      48      48
dbheap           128     128     128     128     128
sortheap         256     256     256     256     256
maxlocks         22      22      22      22      22
stmtheap         1024    1024    1024    1024    1024
SQL STMT         AVERAGE TIMINGS (seconds)
  01             01.34   01.34   01.35   01.35   01.36
  10             02.15   02.00   01.55   01.24   01.00
  15             00.22   00.22   00.22   00.22   00.22
  20             00.84   00.84   00.84   00.84   00.84
  99             00.03   00.03   00.03   00.03   00.03
```

Abbildung 32. Beispielbericht zu den vom Vergleichstest ermittelten Ausführungszeiten

Anmerkung: Die Daten im gezeigten Bericht dienen nur der Illustration. Sie stellen **keine** durch Tests ermittelten Ergebnisse dar.

Zugehörige Konzepte:

- „Durchführen von Vergleichstests" auf Seite 357
- „Erstellung von Vergleichstests" auf Seite 360
- „Ausführung von Vergleichstests" auf Seite 366

Kapitel 13. Konfigurieren von DB2

Konfigurationsparameter

Bei der Erstellung eines Exemplars oder einer Datenbank mit DB2 Universal Database™ wird eine entsprechende Konfigurationsdatei mit Standardparameterwerten erstellt. Diese Parameterwerte können zur Verbesserung der Leistung sowie zur Anpassung anderer Aspekte des Exemplars bzw. der Datenbank modifiziert werden.

Konfigurationsdateien enthalten Parameter, die Werte definieren, wie zum Beispiel die den DB2 UDB-Produkten und einzelnen Datenbanken zugeordneten Ressourcen und die Diagnoseebene. Es gibt zwei Arten von Konfigurationsdateien:

- Die Konfigurationsdatei des Datenbankmanagers für jedes DB2 UDB-Exemplar
- Die Datenbankkonfigurationsdatei für jede einzelne Datenbank

Eine *Konfigurationsdatei des Datenbankmanagers* wird erstellt, wenn ein DB2 UDB-Exemplar erstellt wird. Die in ihr enthaltenen Parameter betreffen Systemressourcen auf Exemplarebene, die von keiner Datenbank abhängig sind, die zu diesem Exemplar gehört. Die Werte vieler dieser Parameter können von den Standardwerten des System 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. Eine Untergruppe der für einen Server verfügbaren Parameter gilt für den Client.

Die Konfigurationsparameter des Datenbankmanagers sind in einer Datei mit dem Namen `db2systm` gespeichert. Diese Datei wird erstellt, wenn das Exemplar des Datenbankmanagers erstellt wird. In auf UNIX basierenden Umgebungen befindet sich diese Datei im Unterverzeichnis `sqllib` für das Exemplar des Datenbankmanagers. Unter Windows finden Sie diese Datei standardmäßig im Unterverzeichnis des Exemplars im Verzeichnis `sqllib`. Wenn die Variable DB2INSTPROF definiert ist, befindet sich die Datei im Unterverzeichnis `instance` des in der Variable DB2INSTPROF angegebenen Verzeichnisses.

In einer Umgebung mit partitionierten Datenbanken befindet sich diese Datei in einem gemeinsam benutzten Dateisystem, so dass alle Datenbankpartitionsserver auf dieselbe Datei zugreifen können. Die Konfiguration des Datenbankmanagers ist auf allen Datenbankpartitionsservern gleich.

Die meisten Parameter haben entweder Einfluss darauf, wie viel Systemressourcen einem einzelnen Datenbankmanagerexemplar 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 diesem Datenbankmanagerexemplar 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 zugehörigen 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.

Parameter für eine einzelne Datenbank werden in einer Konfigurationsdatei namens `SQLDBCON` gespeichert. Diese Datei wird zusammen mit anderen Steuerdateien für die Datenbank im Verzeichnis `SQLnnnnn` 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.

In einer Umgebung mit partitionierten Datenbanken ist für jede Datenbankpartition eine separate Datei `SQLDBCON` vorhanden. Die Werte in der Datei `SQLDBCON` können übereinstimmen oder sich in jeder Datenbankpartition unterscheiden, jedoch ist zu empfehlen, in allen Partitionen die gleichen Werte für die Datenbankkonfigurationsparameter zu verwenden.

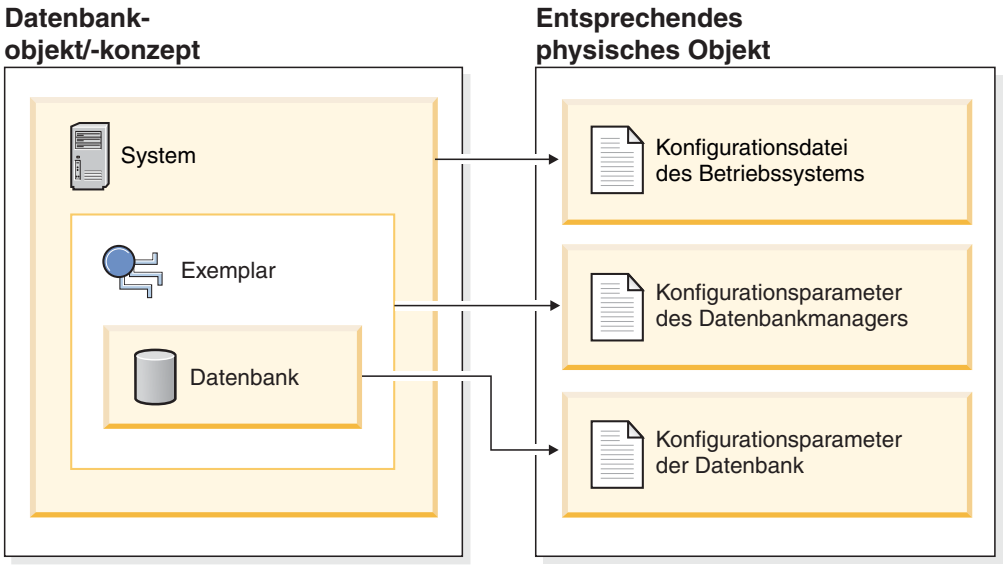


Abbildung 33. Beziehung zwischen Datenbankobjekten und Konfigurationsdateien

Zugehörige Konzepte:

- „Optimierung von Konfigurationsparametern" auf Seite 373

Zugehörige Tasks:

- „Konfigurieren von DB2 mit Konfigurationsparametern" auf Seite 374

Optimierung von Konfigurationsparametern

Der vom Datenbankmanager auf der Basis der Standardwerte 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 vielleicht nicht erreicht.

Da die Standardwerte für Systeme mit relativ kleinem Hauptspeicher, die als Datenbankserver eingesetzt werden, ausgelegt sind, müssen Sie die Parameterwerte möglicherweise ändern, wenn Ihre Umgebung folgende Merkmale aufweist:

- Umfangreiche Datenbanken
- Große Anzahl von Verbindungen
- Hohe Leistungsanforderungen für eine bestimmte Anwendung
- Spezifische Abfrage-/Transaktionsladevorgänge bzw. -arten
- Verschiedene Arten der Konfiguration oder des Einsatzes von Systemen

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 der *Assistent: Konfiguration* bzw. der Befehl AUTOCONFIGURE.

Verschiedene Arten von Anwendungen und Benutzern unterscheiden sich in ihren Anforderungen und Erwartungen an die Antwortzeiten. Die Bandbreite der Anwendungen kann von einfachen Dateneingabebildschirmen bis hin zu strategischen Anwendungen zur Verarbeitung zahlreicher komplexer SQL-Anweisungen mit Zugriff auf Dutzende von Tabellen pro Arbeitseinheit reichen. Zum Beispiel können die Anforderungen an die Antwortzeiten zwischen einer Serviceanwendung für Telefonkunden und einer Stapelanwendung zur Erstellung von Berichten beträchtliche Unterschiede aufweisen.

Einige Konfigurationsparameter können auf den Wert *automatic* gesetzt werden. In diesem Fall werden diese Parameter von DB2® automatisch angepasst, so dass sie die aktuellen Ressourcenanforderungen widerspiegeln.

Zugehörige Konzepte:

- „Konfigurationsparameter" auf Seite 371

Zugehörige Tasks:

- „Konfigurieren von DB2 mit Konfigurationsparametern" auf Seite 374

Zugehörige Referenzen:

- „Konfigurationsparameter - Übersicht" auf Seite 380

Konfigurieren von DB2 mit Konfigurationsparametern

Die Konfigurationsparameter des Datenbankmanagers sind in einer Datei mit dem Namen `db2systm` gespeichert. Die Konfigurationsparameter einer Datenbank sind in einer Datei mit dem Namen `SQLDBCON` gespeichert. Diese Dateien können nicht direkt editiert werden, sondern lediglich mit Hilfe einer bereitgestellten Anwendungsprogrammierschnittstelle (API) oder mit einem Tool, das diese API aufruft, geändert oder angezeigt werden.

Achtung: Wenn Sie die Datei `db2systm` oder `SQLDBCON` mit anderen als den von DB2 vorgesehenen Methoden editieren, können Sie die Datenbank unbrauchbar machen. **Es wird ausdrücklich davon abgeraten**, diese Dateien mit anderen als den von DB2 unterstützten und dokumentierten Methoden zu ändern.

Sie können eine der folgenden Methoden zum Anzeigen, Aktualisieren und Zurücksetzen der Konfigurationsparameter verwenden:

- Über die Steuerzentrale. Das Notizbuch **Exemplar konfigurieren** kann zur Einstellung der Konfigurationsparameter des Datenbankmanagers entweder auf einem Client oder auf einem Server verwendet werden. Das Notizbuch **Datenbank konfigurieren** kann zum Ändern der Werte von Datenbankkonfigurationsparametern verwendet werden. Die DB2-Steuerzentrale stellt auch den Konfigurationsadvisor zum Ändern der Werte der Konfigurationsparameter bereit. Dieser Advisor generiert Werte für Parameter auf der Grundlage Ihrer Antworten zu einer Reihe von Fragen, die zum Beispiel die Auslastung und die Art von Transaktionen betreffen, die mit der Datenbank ausgeführt werden.

  In einer Umgebung mit partitionierten Datenbanken ist die Datei `SQLDBCON` für jede Datenbankpartition vorhanden. Das Notizbuch zum Konfigurieren einer Datenbank ändert den Wert in allen Partitionen, wenn Sie es vom Datenbankobjekt in der Baumstruktursicht der Steuerzentrale aus starten. Wenn Sie das Notizbuch von einem Datenbankpartitionsobjekt aus starten, ändert es nur die Werte für die betreffende Partition. (Es wird jedoch empfohlen, dass die Werte der Konfigurationsparameter in allen Partitionen übereinstimmen.)

  Anmerkung: Der Konfigurationsadvisor ist in Umgebungen mit partitionierten Datenbanken nicht verfügbar.
- Über den Befehlszeilenprozessor. Befehle zum Ändern der Einstellungen können schnell und bequem 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

- Über Anwendungsprogrammierschnittstellen (APIs). APIs können leicht aus einer Anwendung oder einem Programm in einer Hostprogrammiersprache aufgerufen werden.
- Über den Konfigurationsassistenten (für Konfigurationsparameter des Datenbankmanagers). Mit Hilfe des Konfigurationsassistenten können Sie nur die Konfigurationsparameter des Datenbankmanagers auf einem Client einstellen.

Für einige Konfigurationsparameter des Datenbankmanagers muss der Datenbankmanager gestoppt (`db2stop`) und erneut gestartet (`db2start`) werden, um die neuen Parameterwerte in Kraft zu setzen.

Für Datenbankparameter werden Änderungen erst 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.) Bei der Herstellung der ersten neuen Verbindung zur Datenbank werden die Änderungen wirksam.

Andere Parameter können online geändert werden. Diese werden als *online konfigurierbare Konfigurationsparameter* bezeichnet.

Wenn Sie die Einstellung eines online konfigurierbaren Konfigurationsparameters des Datenbankmanagers ändern, während Sie mit einem Exemplar verbunden sind, wendet der Befehl UPDATE DBM CFG die Änderung standardmäßig unverzüglich 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 <exemplarname>
   db2 update dbm cfg using <parametername> <wert>
   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. Sie sollten jedoch beachten, dass einige Parameteränderungen aufgrund des Systemaufwands für die Speicherzuordnung 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 <dbname>
   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 drei 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 an Anweisungsgrenzen und anweisungsähnlichen Grenzen ändern. Wenn Sie zum Beispiel den Wert des Parameters *sortheap* ändern, beginnen alle neuen SQL-Anforderungen mit der Verwendung des neuen Werts.
- **Transaktionsgrenzwert**: Gibt Parameter an, die sich an Transaktionsgrenzen ändern. Zum Beispiel wird ein neuer Wert für den Parameter *dl_expint* nach einer Anweisung COMMIT aktualisiert.

Das Ändern einiger Konfigurationsparameter der Datenbank kann den Zugriffsplan beeinflussen, der vom SQL-Optimierungsprogramm gewählt wird. Nach der Änderung eines dieser Parameter sollten Sie in Betracht ziehen, die Anwendungen erneut zu binden, um sicherzustellen, dass der beste Zugriffsplan für die SQL-Anweisungen verwendet wird. Alle Parameter, die online modifiziert werden (z. B. durch den Befehl UPDATE DATABASE CONFIGURATION IMMEDIATE) veranlassen das SQL-Optimierungsprogramm, neue Zugriffspläne für neue SQL-Anweisungen auszuwählen. Jedoch wird der SQL-Anweisungscache nicht von vorhandenen Einträgen bereinigt. Zur Bereinigung des Inhalts des SQL-Cache verwenden Sie die Anweisung FLUSH PACKAGE CACHE.

Obwohl die neuen Parameterwerte möglicherweise nicht sofort in Kraft treten, werden beim Anzeigen der Parametereinstellungen (mit Hilfe der Befehle GET DATABASE MANAGER CONFIGURATION und GET DATABASE CONFIGURATION) stets die zuletzt aktualisierten Werte angezeigt. Wenn Sie die Klausel SHOW DETAIL in diesen Befehlen zum Anzeigen der Parametereinstellungen angeben, werden sowohl die zuletzt aktualisierten Werte als auch die im Hauptspeicher befindlichen Werte angezeigt.

Anmerkung: Eine Reihe von Konfigurationsparametern (z. B. *userexit*) besitzen laut Beschreibung in der Hilfe und anderer DB2-Dokumentation 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 Konzepte:

- „Konfigurationsparameter" auf Seite 371
- „Optimierung von Konfigurationsparametern" auf Seite 373

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „Konfigurationsparameter - Übersicht" auf Seite 380
- „FLUSH PACKAGE CACHE statement" in *SQL Reference, Volume 2*

Dynamisches Konfigurieren von Parametern

Unter DB2 können Sie eine dynamische Konfiguration nutzen. Sie können bestimmte Konfigurationsparameter in einer DB2-Datenbank oder einem DB2-Exemplar ändern, während die Datenbank aktiv ist, Verbindungen empfängt oder Transaktionen verarbeitet. Das folgende Beispiel erläutert, wie Sie die Funktionen zur dynamischen Konfiguration in DB2 vorteilhaft nutzen können.

In diesem Szenario ist ein Datenbankserver mit einem Speicher von 4 GB konfiguriert, von denen 3,5 GB dem Datenbankmanager zur Verfügung stehen. Es gibt acht Prozessoren. Der Datenbankserver ist für eine einzige Datenbank mit dem Namen DB2DYN dediziert, die im Exemplar DB2INST ausgeführt wird. Die Datenbankauslastung variiert über den Tag und die Woche wie folgt:

- Die Auslastung des Tagesgeschäfts (05:00-20:00) beinhaltet zahlreiche Verbindungen und gleichzeitige Transaktionen.
- Die Auslastung nach Tagesschäftsschluss (20:00-24:00) beinhaltet die Erstellung von Ergebnisberichten und die Ausführung von Entscheidungshilfeabfragen mit einigen wenigen Verbindungen und Transaktionen.
- Die tägliche Systempflegeauslastung (24:00-05:00) beinhaltet Onlineladeoperationen, Teilsicherungen, Indexerstellungen usw.
- Die wöchentliche Systempflegeauslastung beinhaltet umfangreiche Reorganisationen von Tabellen, RUNSTATS-Operationen und die Erstellung größerer Indizes.

In Anbetracht dieser Auslastungsmerkmale könnten Sie das System wie in der folgenden Tabelle gezeigt konfigurieren.

	Tagesgeschäft (05:00 - 20:00)	**Nach Geschäftsschluss (20:00 - 24:00)**	**Tägliche Systempflege (24:00 - 05:00)**	**Wöchentliche Systempflege (sonntags)**
Datenbankaktivität	Hohe Auslastung durch Transaktionen	Entscheidungshilfeabfragen	Laden, Sichern, Indexerstellung	Ausführen von REORG und RUNSTATS in Pufferpool 1
Pufferpool 1 (MB)	1000	500	500	2000
Pufferpool 2 (MB)	1000	500	500	200
Sortierspeicher (MB)	0,1	20	200	200
Katalogcache (MB)	200	200	50	50
Paketcache (MB)	800	200	200	200
Zwischenspeicher für Dienstprogramme (MB)	0	0	1000	0
Diagnosestufe	1	3	4	4

Sie könnten beginnen, indem Sie den Datenbankkonfigurationsparameter *database_memory* auf den Wert 3,5 GB setzen, so dass diese angegebene Größe des verfügbaren Speicherbereichs für die Datenbank reserviert wird. Dies ist eine einmalige Operation, die nach ihrer Ausführung der Datenbank 3,5 GB (bzw. 917.504 4-KB-Seiten) reservierten Speichers für die Pufferpoolerstellung oder für die Konfigurationsanpassung zur Verfügung stellt.

```
db2start
db2 update db cfg for db2dyn using database_memory 917504
db2stop
```

Anschließend könnten Sie die folgenden Prozeduren (scripts) verwenden, um den Übergang der Datenbank von einer Konfiguration zu einer anderen zu vollziehen. (Diese Prozeduren können zur Ausführung an geeigneten Zeitpunkten terminiert werden.)

```
MorningConfiguration.sh

# Dieses Script dient zur Vorbereitung
# des Datenbankservers für die
# morgendliche Konfiguration
# für eine Auslastung bestehend aus
# einer großen Anzahl OLTP-Verbindungen
# und gleichzeitiger Transaktionen.

db2 connect to db2dyn

db2 update db cfg using sortheap 25
db2 update db cfg using util_heap_sz 32
db2 alter bufferpool bufferpool1 size 262144
db2 commit
db2 update db cfg using catcachesz 51200
db2 update db cfg using pkcachesz 204800
db2 alter bufferpool bufferpool2 size 262144
db2 commit
db2 flush package cache dynamic
db2 get db cfg show detail
db2 connect reset

db2 attach to instance db2inst
db2 update dbm cfg using diaglevel 1
db2 get dbm cfg show detail
db2 detach


EveningConfiguration.sh

# Dieses Script dient zur Vorbereitung
# des Datenbankservers für die
# abendliche Konfiguration
# für eine Auslastung bestehend aus
# Entscheidungshilfeabfragen.

db2 connect to db2dyn

db2 alter bufferpool bufferpool1 size 131072
db2 commit
db2 alter bufferpool bufferpool2 size 131072
db2 commit
db2 update db cfg using pkcachesz 51200
db2 update db cfg using catcachesz 51200
db2 update db cfg using util_heap_sz 32
db2 update db cfg using sortheap 5120
db2 flush package cache dynamic
db2 get db cfg show detail
db2 connect reset

db2 attach to instance db2inst
db2 update dbm cfg using diaglevel 3
db2 get dbm cfg show detail
db2 detach


NightTimeConfiguration.sh

# Dieses Script dient zur Vorbereitung
# des Datenbankservers für die
```

```
# tägliche Systempflegekonfiguration
# für eine Auslastung bestehend aus
# Indexerstellungs- sowie Lade- und
# Sicherungsoperationen.

db2 connect to db2dyn

db2 alter bufferpool bufferpool1 size 131072
db2 commit
db2 alter bufferpool bufferpool2 size 131072
db2 commit
db2 update db cfg using catcachesz 51200
db2 update db cfg using pkcachesz 51200
db2 update db cfg using sortheap 51200
db2 update db cfg using util_heap_sz 262144
db2 flush package cache dynamic
db2 get db cfg show detail
db2 connect reset

db2 attach to instance db2inst
db2 update dbm cfg using diaglevel 4
db2 get dbm cfg show detail
db2 detach
```

```
MaintenanceConfiguration.sh

# Dieses Script dient zur Vorbereitung
# des Datenbankservers für die
# wöchentliche Systempflegekonfiguration
# für eine Auslastung bestehend aus
# REORG- und RUNSTATS-Operationen, die in
# Pufferpool 1 ausgeführt werden.

db2 connect to db2dyn

db2 update db cfg using util_heap_sz 32
db2 alter bufferpool bufferpool2 size 51200
db2 commit
db2 update db cfg using sortheap 5120
db2 update db cfg using catcachesz 51200
db2 update db cfg using pkcachesz 51200
db2 alter bufferpool bufferpool1 size 524288
db2 commit
db2 flush package cache dynamic
db2 get db cfg show detail
db2 connect reset

db2 attach to instance db2inst
db2 update dbm cfg using diaglevel 4
db2 get dbm cfg show detail
db2 detach
```

Beachten Sie, dass die Reihenfolge der Operationen von einer Prozedur zur nächsten unterschiedlich ist. Operationen, die den Speicherbedarf senken, sollten vor Operationen ausgeführt werden, die den Speicherbedarf heraufsetzen. Ansonsten könnte das Heraufsetzen fehlschlagen.

Auf alle Änderungen an der Größe von Pufferpools folgt eine COMMIT-Operation, weil Pufferpooländerungen SQL-Operationen und somit Teil der Transaktion sind.

Der Paketcache wird nach jeder Rekonfiguration gelöscht (FLUSH), um dem Optimierungsprogramm zu signalisieren, dass die Konfiguration eine wesentliche Änderung erfahren hat und dass jetzt alle vorhandenen dynamischen SQL-Zugriffspläne ungültig sind.

Dynamische Operationen zur Rekonfiguration des Datenbankmanagers werden ausgeführt, während eine Anwendung mit dem Exemplar verbunden ist. Dynamische Datenbankoperationen werden ausgeführt, während eine Anwendung mit der Datenbank verbunden ist.

Mit der Klausel SHOW DETAIL im Befehl GET DATABASE MANAGER CONFIGURATION oder GET DATABASE CONFIGURATION kann überprüft werden, ob eine dynamische Rekonfigurationsoperation in Kraft getreten ist.

Weitere Datenbankkonfigurationsparameter, die im Rahmen dieses Szenarios ebenfalls hätten dynamisch geändert werden können, sind zum Beispiel:

- Der Parameter *locklist*, dessen Wert dynamisch erhöht werden kann, wenn in der Datenbank häufige Sperreneskalationen auftreten. Der Wert dieses Parameters lässt sich jedoch nicht dynamisch verringern.
- Der Parameter *dft_queryopt*, der nur für neue Verbindungen gültig ist und zur Erhöhung des Optimierungsgrades vor dem Start der Entscheidungshilfeauslastung verwendet werden kann.
- Der Parameter *dft_degree*, der ebenfalls nur für neue Verbindungen gilt und dazu verwendet werden kann, Entscheidungshilfeabfragen mit zusätzlicher abfrageinterner Parallelität zu unterstützen.

Konfigurationsparameter - Übersicht

Übersicht über die Konfigurationsparameter des Datenbankmanagers

In der folgenden Tabelle sind die Parameter der Konfigurationsdatei des Datenbankmanagers für Datenbankserver aufgeführt. Beachten Sie beim Ändern der Konfigurationsparameter des Datenbankmanagers die detaillierten Informationen zu jedem Parameter. Informationen zu spezifischen Betriebsumgebungen mit Standardwerten sind in jeder Parameterbeschreibung enthalten.

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* bezeichnet. Wenn Sie die Einstellung eines online konfigurierbaren Konfigurationsparameters des Datenbankmanagers ändern, während Sie mit einem Exemplar verbunden sind, wendet der Befehl UPDATE DBM CFG die Änderung standardmäßig unverzüglich an. 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 DATABASE MANAGER CONFIGURATION unterstützt. Wenn Sie einen Parameter auf den Wert „AUTOMATIC” setzen, passt DB2 den Parameter automatisch so an, dass er die aktuellen Ressourcenanforderungen widerspiegelt.

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 unter Umständen 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 41. Konfigurierbare Konfigurationsparameter des Datenbankmanagers

Parameter	Onl. Cfg.	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 411
agentpri	Nein	Nein	Hoch	SQLF_KTN_AGENTPRI	26	Sint16	„agentpri - Agentenpriorität" auf Seite 442
aslheapsz	Nein	Nein	Hoch	SQLF_KTN_ASLHEAPSZ	15	Uint32	„aslheapsz - Zwischenspeichergröße für Anwendungsunterstützungsebene" auf Seite 421
audit_buf_sz	Nein	Nein	Hoch	SQLF_KTN_AUDIT_BUF_SZ	312	Sint32	„audit_buf_sz - Prüfpuffergröße" auf Seite 426
authentication[1]	Nein	Nein	Niedrig	SQLF_KTN_AUTHENTICATION	78	Uint16	„authentication - Authentifizierungstyp" auf Seite 538
catalog_noauth	Ja	Nein	Keine	SQLF_KTN_CATALOG_NOAUTH	314	Uint16	„catalog_noauth - Katalogisieren ohne Berechtigung zulässig" auf Seite 539
clnt_krb_plugin	Nein	Nein	Keine	SQLF_KTN_CLNT_KRB_PLUGIN	812	char(33)	„clnt_krb_plugin - Client-Kerberos-Plug-in" auf Seite 539
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 540
comm_bandwidth	Ja	Nein	Mittel	SQLF_KTN_COMM_BANDWIDTH	307	float	„comm_bandwidth - Kommunikationsbandbreite" auf Seite 529
conn_elapse	Ja	Nein	Mittel	SQLF_KTN_CONN_ELAPSE	508	Uint16	„conn_elapse - Antwortzeit für Verbindung" auf Seite 515
cpuspeed	Ja	Nein	Low[2]	SQLF_KTN_CPUSPEED	42	float	„cpuspeed - CPU-Geschwindigkeit" auf Seite 530
datalinks	Nein	Nein	Niedrig	SQLF_KTN_DATALINKS	603	Sint16	„datalinks - Data Links-Unterstützung aktivieren" auf Seite 496

Tabelle 41. Konfigurierbare Konfigurationsparameter des Datenbankmanagers (Forts.)

Parameter	Onl. Cfg.	Auto.	Leist.-Relev.	Token	Token-wert	Datentyp	Zusätzliche Informationen
dft_account_str	Ja	Nein	Keine	SQLF_KTN_DFT_ACCOUNT_STR	28	char(25)	„dft_account_str - Standard-zeichenfolge für Abrechnung" auf Seite 531
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[3] • 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_ TIMES-TAMP • SQLF_KTN_DFT_MON_UOW	29 • 33 • 34 • 35 • 31 • 32 • 36 • 30	Uint16 • Uint16 • Uint16 • Uint16 • Uint16 • Uint16 • Uint16 • Uint16	„dft_monswitches - Monitor-schalter des Standarddatenbanksystems" auf Seite 527
dftdbpath	Ja	Nein	Keine	SQLF_KTN_DFTDBPATH	27	char(215)	„dftdbpath - Standarddatenbankpfad" auf Seite 540
diaglevel	Ja	Nein	Niedrig	SQLF_KTN_DIAGLEVEL	64	Uint16	„diaglevel - Aufzeichnungsebene bei Fehlerdiagnose" auf Seite 524
diagpath	Ja	Nein	Keine	SQLF_KTN_DIAGPATH	65	char(215)	„diagpath - Verzeichnispfad für Diagnosedaten" auf Seite 524
dir_cache	Nein	Nein	Mittel	SQLF_KTN_DIR_CACHE	40	Uint16	„dir_cache - Verzeichniscacheunterstützung" auf Seite 426
discover[4]	Nein	Nein	Mittel	SQLF_KTN_DISCOVER	304	Uint16	„discover - Discovery-Modus" auf Seite 513
discover_inst	Ja	Nein	Niedrig	SQLF_KTN_DISCOVER_INST	308	Uint16	„discover_inst - Discovery-Serverexemplar" auf Seite 514
fcm_num_anchors	Ja	Ja	Mittel	SQLF_KTN_FCM_NUM_ANCHORS	506	Sint32	„fcm_num_anchors - Anzahl von FCM-Nachrichtenankern" auf Seite 516
fcm_num_buffers	Ja	Ja	Mittel	SQLF_KTN_FCM_NUM_BUFFERS	503	Uint32	„fcm_num_buffers - Anzahl FCM-Puffer" auf Seite 517
fcm_num_connect	Ja	Ja	Mittel	SQLF_KTN_FCM_NUM_CONNECT	505	Sint32	„fcm_num_connect - Anzahl von FCM-Verbindungseinträgen" auf Seite 518
fcm_num_rqb	Ja	Ja	Mittel	SQLF_KTN_FCM_NUM_RQB	504	Uint32	„fcm_num_rqb - Anzahl von FCM-Anforderungsblöcken" auf Seite 519
fed_noauth	Ja	Nein	Keine	SQLF_KTN_FED_NOAUTH	806	Uint16	„fed_noauth - Authentifizierung bei Servern mit zusammenge-schlossenen Datenbanken umgehen" auf Seite 542
federated	Nein	Nein	Mittel	SQLF_KTN_FEDERATED	604	Sint16	„federated - Unterstützung für Systeme mit zusammengeschlosse-nen Datenbanken" auf Seite 532
fenced_pool	Nein	Nein	Mittel	SQLF_KTN_FENCED_POOL	80	Sint32	„fenced_pool - Maximale Anzahl abgeschirmter Prozesse" auf Seite 453
group_plugin	Nein	Nein	Keine	SQLF_KTN_GROUP_PLUGIN	810	char(33)	„group_plugin - Gruppen-Plug-in" auf Seite 542
health_mon	Ja	Nein	Niedrig	SQLF_KTN_HEALTH_MON	804	Uint16	„health_mon - Überwachung mit dem Diagnosemonitor" auf Seite 525
indexrec[5]	Ja	Nein	Mittel	SQLF_KTN_INDEXREC	20	Uint16	„indexrec - Zeitpunkt für Indexneuerstellung" auf Seite 483
instance_memory	Nein	Ja	Mittel	SQLF_KTN_INSTANCE_MEMORY	803	Uint64	„instance_memory - Exemplarspeicher" auf Seite 428
intra_parallel	Nein	Nein	Hoch	SQLF_KTN_INTRA_PARALLEL	306	Sint16	„intra_parallel - Partitionsinterne Parallelität aktivieren" auf Seite 522

Tabelle 41. Konfigurierbare Konfigurationsparameter des Datenbankmanagers (Forts.)

Parameter	Onl. Cfg.	Auto.	Leist.-Relev.	Token	Token-wert	Datentyp	Zusätzliche Informationen
java_heap_sz	Nein	Nein	Hoch	SQLF_KTN_JAVA_HEAP_SZ	310	Sint32	„java_heap_sz - Maximale Zwischenspeichergröße für Java-Interpreter" auf Seite 429
jdk_path	Nein	Nein	Keine	SQLF_KTN_JDK_PATH	311	char(255)	„jdk_path - Installationspfad für das Software Developer's Kit für Java" auf Seite 532
keepfenced	Nein	Nein	Mittel	SQLF_KTN_KEEPFENCED	81	Uint16	„keepfenced - Abgeschirmten Prozess beibehalten" auf Seite 454
local_gssplugin	Nein	Nein	Keine	SQLF_KTN_LOCAL_GSSPLUGIN	816	char(33)	„local_gssplugin - GSS-API-Plug-in für lokale Berechtigung auf Exemplarebene" auf Seite 543
max_connections	Nein	Nein	Mittel	SQLF_DBTN_MAX_CONNECTIONS	802	Sint32	„max_connections - Maximale Anzahl von Clientverbindungen" auf Seite 444
max_connretries	Ja	Nein	Mittel	SQLF_KTN_MAX_CONNRETRIES	509	Uint16	„max_connretries - Anzahl Wiederholungen für Verbindungsaufbau zum Knoten" auf Seite 520
max_coordagents	Nein	Nein	Mittel	SQLF_KTN_MAX_COORDAGENTS	501	Sint32	„max_coordagents - Maximale Anzahl koordinierender Agenten" auf Seite 445
max_querydegree	Ja	Nein	Hoch	SQLF_KTN_MAX_QUERYDEGREE	303	Sint32	„max_querydegree - Maximaler Grad der Parallelität bei Abfragen" auf Seite 522
max_time_diff	Nein	Nein	Mittel	SQLF_KTN_MAX_TIME_DIFF	510	Uint16	„max_time_diff - Maximale Zeitdifferenz zwischen Knoten" auf Seite 520
maxagents	Nein	Nein	Mittel	SQLF_KTN_MAXAGENTS	12	Uint32	„maxagents - Maximale Anzahl von Agenten" auf Seite 446
maxcagents	Nein	Nein	Mittel	SQLF_KTN_MAXCAGENTS	13	Sint32	„maxcagents - Maximale Anzahl gleichzeitig aktiver Agenten" auf Seite 448
maxtotfilop	Nein	Nein	Mittel	SQLF_KTN_MAXTOTFILOP	45	Uint16	„maxtotfilop - Maximale Anzahl offener Dateien" auf Seite 450
min_priv_mem	Nein	Nein	Mittel	SQLF_KTN_MIN_PRIV_MEM	43	Uint32	„min_priv_mem - Mindestgröße des reservierten privaten Speichers" auf Seite 413
mon_heap_sz	Nein	Nein	Niedrig	SQLF_KTN_MON_HEAP_SZ	79	Uint16	„mon_heap_sz - Zwischenspeichergröße für Datenbanksystemmonitor" auf Seite 430
nname	Nein	Nein	Keine	SQLF_KTN_NNAME	7	char(8)	„nname - Name der NetBIOS-Workstation" auf Seite 511
notifylevel	Ja	Nein	Niedrig	SQLF_KTN_NOTIFYLEVEL	605	Sint16	„notifylevel - Benachrichtigungsstufe" auf Seite 526
num_initagents	Nein	Nein	Mittel	SQLF_KTN_NUM_INITAGENTS	500	Uint32	„num_initagents - Anfangswert für die Anzahl Agenten im Pool" auf Seite 451
num_initfenced	Nein	Nein	Mittel	SQLF_KTN_NUM_INITFENCED	601	Sint32	„num_initfenced - Anfangswert für die Anzahl abgeschirmter Prozesse" auf Seite 455
num_poolagents	Nein	Nein	Hoch	SQLF_KTN_NUM_POOLAGENTS	502	Sint32	„num_poolagents - Agentenpoolgröße" auf Seite 451
numdb	Nein	Nein	Niedrig	SQLF_KTN_NUMDB	6	Uint16	„numdb - Maximale Anzahl gleichzeitig aktiver Datenbanken einschließlich Host- und iSeries-Datenbanken" auf Seite 533
priv_mem_thresh	Nein	Nein	Mittel	SQLF_KTN_PRIV_MEM_THRESH	44	Sint32	„priv_mem_thresh - Schwellenwert für privaten Speicher" auf Seite 414

Tabelle 41. Konfigurierbare Konfigurationsparameter des Datenbankmanagers (Forts.)

Parameter	Onl. Cfg.	Auto.	Leist.-Relev.	Token	Token-wert	Datentyp	Zusätzliche Informationen
query_heap_sz	Nein	Nein	Mittel	SQLF_KTN_QUERY_HEAP_SZ	49	Sint32	„query_heap_sz - Größe des Abfragezwischenspeichers" auf Seite 415
resync_interval	Nein	Nein	Keine	SQLF_KTN_RESYNC_INTERVAL	68	Uint16	„resync_interval - Intervall für Transaktionsresynchronisation" auf Seite 488
rqrioblk	Nein	Nein	Hoch	SQLF_KTN_RQRIOBLK	1	Uint16	„rqrioblk - E/A-Blockgröße für Clients" auf Seite 424
sheapthres	Nein	Nein	Hoch	SQLF_KTN_SHEAPTHRES	21	Uint32	„sheapthres - Schwellenwert für Sortierspeicher" auf Seite 416
spm_log_file_sz	Nein	Nein	Niedrig	SQLF_KTN_SPM_LOG_FILE_SZ	90	Sint32	„spm_log_file_sz - Protokolldateigröße für SPM" auf Seite 489
spm_log_path	Nein	Nein	Mittel	SQLF_KTN_SPM_LOG_PATH	313	char(226)	„spm_log_path - Protokolldateipfad für SPM" auf Seite 490
spm_max_resync	Nein	Nein	Niedrig	SQLF_KTN_SPM_MAX_RESYNC	91	Sint32	„spm_max_resync - Maximale Anzahl von SPM-Resynchronisationsagenten" auf Seite 490
spm_name	Nein	Nein	Keine	SQLF_KTN_SPM_NAME	92	char(8)	„spm_name - Name des Synchronisationspunktmanagers" auf Seite 491
srvcon_auth	Nein	Nein	Keine	SQLF_KTN_SRVCON_AUTH	815	Uint16	„srvcon_auth - Authentifizierungstyp für ankommende Verbindungen auf dem Server" auf Seite 543
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 544
srv_plugin_mode	Nein	Nein	Keine	SQLF_KTN_SRV_PLUGIN_MODE	809	Uint16	„srv_plugin_mode - Server-Plug-in-Modus" auf Seite 545
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 545
start_stop_time	Ja	Nein	Niedrig	SQLF_KTN_START_STOP_TIME	511	Uint16	„start_stop_time - Zeitlimit für DB2START und DB2STOP" auf Seite 521
svcename	Nein	Nein	Keine	SQLF_KTN_SVCENAME	24	char(14)	„svcename - TCP/IP-Servicename" auf Seite 511
sysadm_group	Nein	Nein	Keine	SQLF_KTN_SYSADM_GROUP	39	char(16)	„sysadm_group - SYSADM-Gruppenname" auf Seite 546
sysctrl_group	Nein	Nein	Keine	SQLF_KTN_SYSCTRL_GROUP	63	char(16)	„sysctrl_group - SYSCTRL-Gruppenname" auf Seite 547
sysmaint_group	Nein	Nein	Keine	SQLF_KTN_SYSMAINT_GROUP	62	char(16)	„sysmaint_group - SYSMAINT-Gruppenname" auf Seite 547
sysmon_group	Nein	Nein	Keine	SQLF_KTN_SYSMON	808	char(9)	„sysmon_group - Berechtigungsgruppenname für Systemmonitor" auf Seite 548
tm_database	Nein	Nein	Keine	SQLF_KTN_TM_DATABASE	67	char(8)	„tm_database - Name für Transaktionsmanagerdatenbank" auf Seite 491
tp_mon_name	Nein	Nein	Keine	SQLF_KTN_TP_MON_NAME	66	char(19)	„tp_mon_name - Name des Transaktionsprozessormonitors" auf Seite 534
tpname	Nein	Nein	Keine	SQLF_KTN_TPNAME	25	char(64)	„tpname - APPC-Transaktionsprogrammname" auf Seite 512

Tabelle 41. Konfigurierbare Konfigurationsparameter des Datenbankmanagers (Forts.)

Parameter	Onl. Cfg.	Auto.	Leist.-Relev.	Token	Token-wert	Datentyp	Zusätzliche Informationen
trust_allclnts[6]	Nein	Nein	Keine	SQLF_KTN_TRUST_ALLCLNTS	301	Uint16	„trust_allclnts - Alle Clients akzeptieren" auf Seite 549
trust_clntauth	Nein	Nein	Keine	SQLF_KTN_TRUST_CLNTAUTH	302	Uint16	„trust_clntauth - Authentifizierung gesicherter Clients" auf Seite 550
use_sna_auth	Ja	Nein	Keine	SQLF_KTN_USE_SNA_AUTH	805	Uint16	„use_sna_auth - SNA-Authentifizierung verwenden" auf Seite 551
util_impact_lim	Ja	Nein	Hoch	SQLF_KTN_UTIL_IMPACT_LIM	807	Uint32	„util_impact_lim - Richtlinie für Exemplarauslastungswirkung" auf Seite 536

Anmerkungen:

1. Gültige Werte (in `sqlenv.h` definiert) sind:

```
SQL_AUTHENTICATION_SERVER (0)
SQL_AUTHENTICATION_CLIENT (1)
SQL_AUTHENTICATION_DCS (2)
SQL_AUTHENTICATION_DCE (3)
SQL_AUTHENTICATION_SVR_ENCRYPT (4)
SQL_AUTHENTICATION_DCS_ENCRYPT (5)
SQL_AUTHENTICATION_DCE_SVR_ENC (6)
SQL_AUTHENTICATION_KERBEROS (7)
SQL_AUTHENTICATION_KRB_SVR_ENC (8)
SQL_AUTHENTICATION_GSSPLUGIN (9)
SQL_AUTHENTICATION_GSS_SVR_ENC (10)
SQL_AUTHENTICATION_DATAENC (11)
SQL_AUTHENTICATION_DATAENC_CMP (12)
SQL_AUTHENTICATION_NOT_SPEC (255)
```

2. Der Parameter *cpuspeed* kann sich wesentlich auf die Leistung auswirken, jedoch sollten Sie den Standardwert verwenden, sofern nicht spezielle Umstände vorliegen, wie sie in der Parameterbeschreibung dokumentiert sind.
3.
```
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
```

4. Gültige Werte (in `sqlutil.h` definiert) sind:

```
SQLF_DSCVR_KNOWN (1)
SQLF_DSCVR_SEARCH (2)
```

5. Gültige Werte (in `sqlutil.h` definiert) sind:

```
SQLF_INX_REC_SYSTEM (0)
SQLF_INX_REC_REFERENCE (1)
```

6. Gültige Werte (in `sqlutil.h` definiert) sind:

```
SQLF_TRUST_ALLCLNTS_NO (0)
SQLF_TRUST_ALLCLNTS_YES (1)
SQLF_TRUST_ALLCLNTS_DRDAONLY (2)
```

Tabelle 42. Informative Konfigurationsparameter des Datenbankmanagers

Parameter	Token	Token-wert	Daten-typ	Zusätzliche Informationen
nodetype[1]	SQLF_KTN_NODETYPE	100	Uint16	„nodetype - Knotenart des Systems" auf Seite 533
release	SQLF_KTN_RELEASE	101	Uint16	„release - Releasestand der Datenbankkonfiguration" auf Seite 495

Tabelle 42. Informative Konfigurationsparameter des Datenbankmanagers (Forts.)

Parameter	Token	Token-wert	Daten-typ	Zusätzliche Informationen

Anmerkungen:
1. Gültige Werte (in `sqlutil.h` definiert) sind:
`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.

Die Spalte „Auto." in der folgenden Tabelle gibt an, ob der Parameter das Schlüsselwort AUTOMATIC im Befehl UPDATE DATABASE CONFIGURATION unterstützt. Wenn Sie einen Parameter auf den Wert „AUTOMATIC" setzen, passt DB2 den Parameter automatisch so an, dass er die aktuellen Ressourcenanforderungen widerspiegelt.

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 unter Umständen 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 43. Konfigurierbare Konfigurationsparameter für die Datenbank

Parameter	Onl. Cfg.	Auto.	Leist.-Relev.	Token	Token-wert	Datentyp	Zusätzliche Informationen
alt_collate	Nein	Nein	Keine	SQLF_DBTN_ALT_COLLATE	809	Uint32	„alt_collate - Alternative Sortierfolge" auf Seite 493
app_ctl_heap_sz	Nein	Nein	Mittel	SQLF_DBTN_APP_CTL_HEAP_SZ	500	Uint16	„app_ctl_heap_sz - Zwischenspeichergröße für Anwendungssteuerung" auf Seite 406
appgroup_mem_sz	Nein	Nein	Mittel	SQLF_DBTN_APPGROUP_MEM_SZ	800	Uint32	„appgroup_mem_sz - Maximale Speichergröße für Anwendungsgruppe" auf Seite 409
applheapsz	Nein	Nein	Mittel	SQLF_DBTN_APPLHEAPSZ	51	Uint16	„applheapsz - Zwischenspeichergröße für Anwendungen" auf Seite 412
archretrydelay	Ja	Nein	Keine	SQLF_DBTN_ARCHRETRYDELAY	828	Uint16	„archretrydelay - Wiederholungsintervall für Archivierung bei Fehler" auf Seite 467
autonomic_switches • *auto_maint* • *auto_db_backup* • *auto_tbl_maint* • *auto_runstats* • *auto_stats_prof* • *auto_prof_upd* • *auto_reorg*	Ja	Nein	Mittel	SQLF_DBTN_AUTONOMIC_SWITCHES[1] • SQLF_ENABLE_AUTO_MAINT • SQLF_ENABLE_AUTO_DB_BACKUP • SQLF_ENABLE_AUTO_TBL_MAINT • SQLF_ENABLE_AUTO_RUNSTATS • SQLF_ENABLE_AUTO_STATS_PROF • SQLF_ENABLE_AUTO_PROF_UPD • SQLF_ENABLE_AUTO_REORG	830 • 831 • 833 • 835 • 837 • 839 • 844 • 841	Uint32	„autonomic_switches - Schalter für automatische Verwaltung" auf Seite 508
autorestart	Ja	Nein	Niedrig	SQLF_DBTN_AUTO_RESTART	25	Uint16	„autorestart - Automatischer Neustart aktiviert" auf Seite 477
avg_appls	Ja	Nein	Hoch	SQLF_DBTN_AVG_APPLS	47	Uint16	„avg_appls - Durchschnittliche Anzahl aktiver Anwendungen" auf Seite 443
blk_log_dsk_ful	Ja	Nein	Keine	SQLF_DBTN_BLK_LOG_DSK_FUL	804	Uint16	„blk_log_dsk_ful - Bei voller Protokollplatte blockieren" auf Seite 467
catalogcache_sz	Ja	Nein	Hoch	SQLF_DBTN_CATALOGCACHE_SZ	56	Sint32	„catalogcache_sz - Katalogcachegröße" auf Seite 396
chngpgs_thresh	Nein	Nein	Hoch	SQLF_DBTN_CHNGPGS_THRESH	38	Uint16	„chngpgs_thresh - Schwellenwert für geänderte Seiten" auf Seite 435
database_memory	Nein	Ja	Mittel	SQLF_DBTN_DATABASE_MEMORY	803	Uint64	„database_memory - Größe des gemeinsam benutzten Datenbankspeichers" auf Seite 398
dbheap	Ja	Nein	Mittel	SQLF_DBTN_DB_HEAP	58	Uint64	„dbheap - Zwischenspeicher für Datenbank" auf Seite 399
dft_degree	Ja	Nein	Hoch	SQLF_DBTN_DFT_DEGREE	301	Sint32	„dft_degree - Standardgrad der Parallelität" auf Seite 502
dft_extent_sz	Ja	Nein	Mittel	SQLF_DBTN_DFT_EXTENT_SZ	54	Uint32	„dft_extent_sz - Standardwert für EXTENTSIZE bei Tabellenbereichen" auf Seite 436

Tabelle 43. Konfigurierbare Konfigurationsparameter für die Datenbank (Forts.)

Parameter	Onl. Cfg.	Auto.	Leist.-Relev.	Token	Token-wert	Datentyp	Zusätzliche Informationen
dft_loadrec_ses	Ja	Nein	Mittel	SQLF_DBTN_DFT_LOADREC_SES	42	Sint16	„dft_loadrec_ses - Standard-anzahl von Sitzungen für Wiederherstellung" auf Seite 478
dft_mttb_types	Nein	Nein	Keine	SQLF_DBTN_DFT_MTTB_TYPES	843	Uint32	„dft_mttb_types - Standard-typen von verwalteten Tabellen zur Optimierung" auf Seite 503
dft_prefetch_sz	Ja	Ja	Mittel	SQLF_DBTN_DFT_PREFETCH_SZ	40	Sint16	„dft_prefetch_sz - Standard-wert für PREFETCHSIZE" auf Seite 436
dft_queryopt	Ja	Nein	Mittel	SQLF_DBTN_DFT_QUERYOPT	57	Sint32	"dft_queryopt-Standard-abfrageoptimierungs-klasse"
dft_refresh_age	Nein	Nein	Mittel	SQLF_DBTN_DFT_REFRESH_AGE	702	char(22)	„dft_refresh_age - Standardaktualisierungsalter" auf Seite 504
dft_sqlmathwarn	Nein	Nein	Keine	SQLF_DBTN_DFT_SQLMATHWARN	309	Sint16	„dft_sqlmathwarn - Bei arithmetischen Ausnahme-bedingungen fortsetzen" auf Seite 504
discover_db	Ja	Nein	Mittel	SQLF_DBTN_DISCOVER	308	Uint16	„discover_db - Discovery-Unterstützung für diese Datenbank" auf Seite 514
dl_expint	Ja	Nein	Keine	SQLF_DBTN_DL_EXPINT	350	Sint32	„dl_expint - Ablaufintervall für Data Links-Zugriffs-Token" auf Seite 496
dl_num_copies	Ja	Nein	Keine	SQLF_DBTN_DL_NUM_COPIES	351	Uint16	„dl_num_copies - Anzahl Data Links-Kopien" auf Seite 497
dl_time_drop	Ja	Nein	Keine	SQLF_DBTN_DL_TIME_DROP	353	Uint16	„dl_time_drop - Data Links-Zeit nach DROP" auf Seite 497
dl_token	Ja	Nein	Niedrig	SQLF_DBTN_DL_TOKEN	602	char(10)	„dl_token - Data Links-Tokenalgorithmus" auf Seite 498
dl_upper	Ja	Nein	Keine	SQLF_DBTN_DL_UPPER	603	Sint16	„dl_upper - Data Links-Token in Großbuchstaben" auf Seite 498
dl_wt_iexpint	Ja	Nein	Keine	SQLF_DBTN_DL_WT_IEXPINT	354	Sint32	„dl_wt_iexpint - Verfalls-intervall für Data Links-Schreibtoken" auf Seite 499
dlchktime	Ja	Nein	Mittel	SQLF_DBTN_DLCHKTIME	9	Uint32	„dlchktime - Zeitintervall für Prüfung gegenseitiger Sperren" auf Seite 431
dyn_query_mgmt	Nein	Nein	Niedrig	SQLF_DBTN_DYN_QUERY_MGMT	604	Uint16	„dyn_query_mgmt - Dyna-mische SQL-Abfrageverwaltung" auf Seite 492
estore_seg_sz	Nein	Nein	Mittel	SQLF_DBTN_ESTORE_SEG_SZ	303	Sint32	„estore_seg_sz - Segment-größe für erweiterten Speicher" auf Seite 438
failarchpath	Ja	Nein	Keine	SQLF_DBTN_FAILARCHPATH	826	char(243)	„failarchpath - Protokoll-archivpfad für Funktionsübernahme" auf Seite 468
groupheap_ratio	Nein	Nein	Mittel	SQLF_DBTN_GROUPHEAP_RATIO	801	Uint16	"groupheap_ratio - Für Anwendungsgruppen-zwischenspeicher vorgesehener Speicher in Prozent"

Tabelle 43. Konfigurierbare Konfigurationsparameter für die Datenbank (Forts.)

Parameter	Onl. Cfg.	Auto.	Leist.-Relev.	Token	Token-wert	Datentyp	Zusätzliche Informationen
hadr_local_host	Nein	Nein	Keine	SQLF_DBTN_HADR_LOCAL_HOST	811	char(256)	„hadr_local_host - Name des lokalen Hosts für HADR" auf Seite 479
hadr_local_svc	Nein	Nein	Keine	SQLF_DBTN_HADR_LOCAL_SVC	812	char(41)	„hadr_local_svc - Lokaler HADR-Servicename" auf Seite 480
hadr_remote_host	Nein	Nein	Keine	SQLF_DBTN_HADR_REMOTE_HOST	813	char(256)	„hadr_remote_host - Name des fernen Hosts für HADR" auf Seite 480
hadr_remote_inst	Nein	Nein	Keine	SQLF_DBTN_HADR_REMOTE_INST	815	char(9)	„hadr_remote_inst - HADR-Exemplarname des fernen Servers" auf Seite 481
hadr_remote_svc	Nein	Nein	Keine	SQLF_DBTN_HADR_REMOTE_SVC	814	char(41)	„hadr_remote_svc - Ferner HADR-Servicename" auf Seite 481
hadr_syncmode	Nein	Nein	Keine	SQLF_DBTN_HADR_SYNCMODE	817	Uint32	„hadr_syncmode - HADR-Synchronisationsmodus für Protokollierung im Peerstatus" auf Seite 481
hadr_timeout	Nein	Nein	Keine	SQLF_DBTN_HADR_TIMEOUT	816	Sint32	„hadr_timeout - HADR-Zeitlimitwert" auf Seite 482
indexrec[2]	Ja	Nein	Mittel	SQLF_DBTN_INDEXREC	30	Uint16	„indexrec - Zeitpunkt für Indexneuerstellung" auf Seite 483
locklist	Ja	Nein	Hoch bei Einfluss auf die Eskalation	SQLF_DBTN_LOCK_LIST	704	Uint64	„locklist - Maximaler Speicher für Sperrenliste" auf Seite 399
locktimeout	Nein	Nein	Mittel	SQLF_DBTN_LOCKTIMEOUT	34	Sint16	„locktimeout - Zeitlimit für Sperren" auf Seite 432
logarchmeth1	Ja	Nein	Keine	SQLF_DBTN_LOGARCHMETH1	822	Uint16	"logarchmeth1 - Primäre Protokollarchivierungs-methode"
logarchmeth2	Ja	Nein	Keine	SQLF_DBTN_LOGARCHMETH2	823	Uint16	"logarchmeth2 - Sekundäre Protokollarchivierungs-methode"
logarchopt1	Ja	Nein	Keine	SQLF_DBTN_LOGARCHOPT1	824	char(243)	„logarchopt1 - Optionen für primäres Protokollarchiv" auf Seite 470
logarchopt2	Ja	Nein	Keine	SQLF_DBTN_LOGARCHOPT2	825	char(243)	„logarchopt2 - Optionen für sekundäres Protokollarchiv" auf Seite 470
logbufsz	Nein	Nein	Hoch	SQLF_DBTN_LOGBUFSZ	33	Uint16	„logbufsz - Protokollpuffergröße" auf Seite 402
logfilsiz	Nein	Nein	Mittel	SQLF_DBTN_LOGFIL_SIZ	92	Uint32	„logfilsiz - Protokolldateigröße" auf Seite 456
logindexbuild	Ja	Ja	Keine	SQLF_DBTN_LOGINDEXBUILD	818	Uint32	„logindexbuild - Erstellte Indexseiten protokollieren" auf Seite 470
logprimary	Nein	Nein	Mittel	SQLF_DBTN_LOGPRIMARY	16	Uint16	„logprimary - Anzahl primärer Protokolldateien" auf Seite 458
logretain[3]	Nein	Nein	Niedrig	SQLF_DBTN_LOG_RETAIN	23	Uint16	„logretain - Beibehalten von Protokollen aktivieren" auf Seite 471
logsecond	Ja	Nein	Mittel	SQLF_DBTN_LOGSECOND	17	Uint16	„logsecond - Anzahl sekundärer Protokolldateien" auf Seite 460
max_log	Ja	Ja		SQLF_DBTN_MAX_LOG	807	Uint16	„max_log - Maximuale Protokollgröße pro Transaktion" auf Seite 462

Tabelle 43. Konfigurierbare Konfigurationsparameter für die Datenbank (Forts.)

Parameter	Onl. Cfg.	Auto.	Leist.-Relev.	Token	Token-wert	Datentyp	Zusätzliche Informationen
maxappls	Ja	Ja	Mittel	SQLF_DBTN_MAXAPPLS	6	Uint16	„maxappls - Maximale Anzahl aktiver Anwendungen" auf Seite 447
maxfilop	Ja	Nein	Mittel	SQLF_DBTN_MAXFILOP	3	Uint16	„maxfilop - Maximale Anzahl offener Datenbank-dateien pro Anwendung" auf Seite 449
maxlocks	Ja	Nein	Hoch bei Einfluss auf die Eskalation	SQLF_DBTN_MAXLOCKS	15	Uint16	„maxlocks - Maximale Anzahl Sperren pro Anwendung" auf Seite 433
min_dec_div_3	Nein	Nein	Hoch	SQLF_DBTN_MIN_DEC_DIV_3	605	Sint32	„min_dec_div_3 - 3 Komma-stellen bei Dezimaldivision" auf Seite 423
mincommit	Ja	Nein	Hoch	SQLF_DBTN_MINCOMMIT	32	Uint16	„mincommit - Anzahl der Gruppenfestschreibungen" auf Seite 471
mirrorlogpath	Nein	Nein	Niedrig	SQLF_DBTN_MIRRORLOGPATH	806	char(242)	„mirrorlogpath - Pfad für Protokollspiegelung" auf Seite 462
newlogpath	Nein	Nein	Niedrig	SQLF_DBTN_NEWLOGPATH	20	char(242)	„newlogpath - Datenbank-protokollpfad ändern" auf Seite 463
num_db_backups	Ja	Nein	Keine	SQLF_DBTN_NUM_DB_BACKUPS	601	Uint16	„num_db_backups - Anzahl der Datenbanksicherungen" auf Seite 484
num_estore_segs	Nein	Nein	Mittel	SQLF_DBTN_NUM_ESTORE_SEGS	304	Sint32	„num_estore_segs - Segmentanzahl für erweiter-ten Speicher" auf Seite 438
num_freqvalues	Ja	Nein	Niedrig	SQLF_DBTN_NUM_FREQVALUES	36	Uint16	„num_freqvalues - Anzahl der häufigsten Werte" auf Seite 506
num_iocleaners	Nein	Nein	Hoch	SQLF_DBTN_NUM_IOCLEANERS	37	Uint16	„num_iocleaners - Anzahl asynchroner Seitenlöschfunktionen" auf Seite 439
num_ioservers	Nein	Nein	Hoch	SQLF_DBTN_NUM_IOSERVERS	39	Uint16	„num_ioservers - Anzahl von E/A-Servern" auf Seite 440
num_log_span	Ja	Ja		SQLF_DBTN_NUM_LOG_SPAN	808	Uint16	„num_log_span - Anzahl verwendeter Protokolldateien" auf Seite 465
num_quantiles	Ja	Nein	Niedrig	SQLF_DBTN_NUM_QUANTILES	48	Uint16	„num_quantiles - Anzahl der Quantile für Spalten" auf Seite 507
numarchretry	Ja	Nein	Keine	SQLF_DBTN_NUMARCHRETRY	827	Uint16	„numarchretry - Anzahl der Wiederholungen bei Fehler" auf Seite 473
overflowlogpath	Nein	Nein	Mittel	SQLF_DBTN_OVERFLOWLOGPATH	805	char(242)	„overflowlogpath - Überlaufprotokollpfad" auf Seite 465
pckcachesz	Ja	Nein	Hoch	SQLF_DBTN_PCKCACHE_SZ	505	Uint32	„pckcachesz - Größe des Paketcache" auf Seite 403
rec_his_retentn	Nein	Nein	Keine	SQLF_DBTN_REC_HIS_RETENTN	43	Sint16	„rec_his_retentn - Auf-bewahrungszeitraum für Wiederherstellungsprotokoll" auf Seite 485
seqdetect	Ja	Nein	Hoch	SQLF_DBTN_SEQDETECT	41	Uint16	„seqdetect - Markierung für Sequenzerkennung" auf Seite 441

Tabelle 43. Konfigurierbare Konfigurationsparameter für die Datenbank (Forts.)

Parameter	Onl. Cfg.	Auto.	Leist.-Relev.	Token	Token-wert	Datentyp	Zusätzliche Informationen
sheapthres_shr	Nein	Nein	Hoch	SQLF_DBTN_SHEAPTHRES_SHR	802	Uint32	„sheapthres_shr - Schwellenwert für Sortierspeicher für gemeinsame Sortiervorgänge" auf Seite 405
softmax	Nein	Nein	Mittel	SQLF_DBTN_SOFTMAX	5	Uint16	„softmax - Vor bedingtem Prüfpunkt zu schreibende Protokollsätze" auf Seite 474
sortheap	Ja	Nein	Hoch	SQLF_DBTN_SORT_HEAP	52	Uint32	„sortheap - Zwischenspeichergröße für Sortierlisten" auf Seite 418
stat_heap_sz	Nein	Nein	Niedrig	SQLF_DBTN_STAT_HEAP_SZ	45	Uint32	„stat_heap_sz - Größe des Statistikzwischenspeichers" auf Seite 419
stmtheap	Ja	Nein	Mittel	SQLF_DBTN_STMT_HEAP	821	Uint32	"stmtheap - Größe des Anweisungszwischenspeichers"
trackmod	Nein	Nein	Niedrig	SQLF_DBTN_TRACKMOD	703	Uint16	„trackmod - Geänderte Seiten protokollieren" auf Seite 486
tsm_mgmtclass	Ja	Nein	Keine	SQLF_DBTN_TSM_MGMTCLASS	307	char(30)	„tsm_mgmtclass - Tivoli Storage Manager-Verwaltungsklasse" auf Seite 486
tsm_nodename	Ja	Nein	Keine	SQLF_DBTN_TSM_NODENAME	306	char(64)	„tsm_nodename - TSM-Knotenname" auf Seite 487
tsm_owner	Ja	Nein	Keine	SQLF_DBTN_TSM_OWNER	305	char(64)	„tsm_owner - TSM-Eignername" auf Seite 487
tsm_password	Ja	Nein	Keine	SQLF_DBTN_TSM_PASSWORD	501	char(64)	„tsm_password - TSM-Kennwort" auf Seite 488
userexit	Nein	Nein	Niedrig	SQLF_DBTN_USER_EXIT	24	Uint16	„userexit - Benutzerexit aktivieren" auf Seite 475
util_heap_sz	Ja	Nein	Niedrig	SQLF_DBTN_UTIL_HEAP_SZ	55	Uint32	„util_heap_sz - Zwischenspeichergröße für Dienstprogramme" auf Seite 406
vendoropt	Ja	Nein	Keine	SQLF_DBTN_VENDOROPT	829	char(242)	„vendoropt - Lieferantenoptionen" auf Seite 476

Tabelle 43. Konfigurierbare Konfigurationsparameter für die Datenbank (Forts.)

Parameter	Onl. Cfg.	Auto.	Leist.-Relev.	Token	Token-wert	Datentyp	Zusätzliche Informationen

Anmerkungen:

1. 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 x0xx): 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 xx0x xxxx): auto_prof_upd
Bit 7 off (xxxx xxxx x0xx xxxx): auto_reorg
             0    0     1   9
```
   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
             0    0    7    F
```
2. Gültige Werte (in `sqlutil.h` definiert) sind:
```
SQLF_INX_REC_SYSTEM (0)
SQLF_INX_REC_REFERENCE (1)
SQLF_INX_REC_RESTART (2)
```
3. Gültige Werte (in `sqlutil.h` definiert) sind:
```
SQLF_LOGRETAIN_NO (0)
SQLF_LOGRETAIN_RECOVERY (1)
SQLF_LOGRETAIN_CAPTURE (2)
```

Tabelle 44. Informative Konfigurationsparameter für die Datenbank

Parameter	Token	Token-wert	Datentyp	Zusätzliche Informationen
backup_pending	SQLF_DBTN_BACKUP_PENDING	112	Uint16	„backup_pending - Sicherung anstehend" auf Seite 499
codepage	SQLF_DBTN_CODEPAGE	101	Uint16	„codepage - Codepage für die Datenbank" auf Seite 494
codeset	SQLF_DBTN_CODESET	120	char(9)[1]	„codeset - Codierter Zeichensatz für die Datenbank" auf Seite 494
collate_info	SQLF_DBTN_COLLATE_INFO	44	char(260)	„collate_info - Informationen zur Sortierfolge" auf Seite 494
country	SQLF_DBTN_COUNTRY	100	Uint16	„country - Datenbankgebietscode" auf Seite 495
database_consistent	SQLF_DBTN_CONSISTENT	111	Uint16	„database_consistent - Datenbank ist konsistent" auf Seite 500
database_level	SQLF_DBTN_DATABASE_LEVEL	124	Uint16	„database_level - Releasestand der Datenbank" auf Seite 495
hadr_db_role	SQLF_DBTN_HADR_DB_ROLE	810	Uint32	„hadr_db_role - Rolle der HADR-Datenbank" auf Seite 479

Tabelle 44. Informative Konfigurationsparameter für die Datenbank (Forts.)

Parameter	Token	Token-wert	Datentyp	Zusätzliche Informationen
log_retain_status	SQLF_DBTN_LOG_RETAIN_STATUS	114	Uint16	„log_retain_status - Statusanzeiger für Beibehalten der Protokolle" auf Seite 500
loghead	SQLF_DBTN_LOGHEAD	105	char(12)	„loghead - Erste aktive Protokolldatei" auf Seite 458
logpath	SQLF_DBTN_LOGPATH	103	char(242)	„logpath - Pfad zu Protokolldateien" auf Seite 458
multipage_alloc	SQLF_DBTN_MULTIPAGE_ALLOC	506	Uint16	„multipage_alloc - Zuordnung aus mehreren Seiten bestehender Datei aktiv" auf Seite 500
numsegs	SQLF_DBTN_NUMSEGS	122	Uint16	„numsegs - Standardanzahl von SMS-Behältern" auf Seite 441
release	SQLF_DBTN_RELEASE	102	Uint16	„release - Releasestand der Datenbankkonfiguration" auf Seite 495
restore_pending	SQLF_DBTN_RESTORE_PENDING	503	Uint16	„restore_pending - Wiederherstellung anstehend" auf Seite 501
rollfwd_pending	SQLF_DBTN_ROLLFWD_PENDING	113	Uint16	„rollfwd_pending - Aktualisierende Wiederherstellung anstehend" auf Seite 501
territory	SQLF_DBTN_TERRITORY	121	char(5)[2]	„territory - Datenbankgebiet" auf Seite 496
user_exit_status	SQLF_DBTN_USER_EXIT_STATUS	115	Uint16	„user_exit_status - Statusanzeiger für Benutzerexit" auf Seite 501

Anmerkungen:

1. char(17) unter HP-UX und Solaris-Betriebsumgebung.
2. char(33) unter HP-UX und Solaris-Betriebsumgebung.

Übersicht über die Konfigurationsparameter für den DB2-Verwaltungsserver (DAS)

Tabelle 45. DAS-Konfigurationsparameter

Parameter	Parametertyp	Zusätzliche Informationen
authentication	Konfigurierbar	„authentication - DAS-Authentifizierungstyp" auf Seite 552
contact_host	Online konfigurierbar	„contact_host - Speicherposition der Liste mit Ansprechpartnern" auf Seite 553
das_codepage	Online konfigurierbar	„das_codepage - DAS-Codepage" auf Seite 553
das_territory	Online konfigurierbar	„das_territory - DAS-Gebiet" auf Seite 554
dasadm_group	Konfigurierbar	„dasadm_group - DASADM-Gruppenname" auf Seite 554
db2system	Online konfigurierbar	„db2system - Name des DB2-Serversystems" auf Seite 555
discover	Online konfigurierbar	„discover - DAS-Discovery-Modus" auf Seite 555
exec_exp_task	Konfigurierbar	„exec_exp_task - Verfallene Tasks ausführen" auf Seite 556
jdk_64_path	Online konfigurierbar	„jdk_64_path - Installationspfad für Software Developer's Kit (64 Bit) für Java auf DAS" auf Seite 557
jdk_path	Online konfigurierbar	„jdk_path - Installationspfad für Software Developer's Kit für Java auf DAS" auf Seite 557
sched_enable	Konfigurierbar	„sched_enable - Schedulermodus" auf Seite 558
sched_userid	Informativ	„sched_userid - Benutzer-ID für Scheduler" auf Seite 558
smtp_server	Online konfigurierbar	„smtp_server - SMTP-Server" auf Seite 559
toolscat_db	Konfigurierbar	„toolscat_db - Toolskatalogdatenbank" auf Seite 559
toolscat_inst	Konfigurierbar	„toolscat_inst - Exemplar der Toolskatalogdatenbank" auf Seite 560
toolscat_schema	Konfigurierbar	„toolscat_schema - Schema der Toolskatalogdatenbank" auf Seite 561

Einzelheiten zu Parametern nach Funktion

Die folgenden Abschnitte enthalten zusätzliche Einzelinformationen, die Ihnen das Verständnis der Parameter erleichtern und Sie bei der Optimierung der verschiedenen Konfigurationsparameter unterstützen sollen. Die Beschreibungen der einzelnen Parameter sind nach Funktion bzw. Zweck der Parameter geordnet:

- „Kapazitätsverwaltung" auf Seite 395
- „Protokollieren und Wiederherstellung" auf Seite 456
- „Datenbankverwaltung" auf Seite 492
- „Kommunikation" auf Seite 510
- „Umgebung mit partitionierter Datenbank" auf Seite 515
- „Exemplarverwaltung" auf Seite 523
- „DB2-Verwaltungsserver" auf Seite 552

Die Beschreibung der Parameter enthält jeweils folgende Informationen:

Konfigurationstyp
Gibt an, welche Konfigurationsdatei die Einstellung für den Parameter enthält:

- *Datenbankmanager* betrifft ein Exemplar des Datenbankmanagers und alle Datenbanken, die innerhalb dieses Exemplars definiert sind.

- *Datenbank* betrifft eine bestimmte Datenbank.

Parametertyp

Gibt an, ob der Parameterwert geändert werden kann oder nicht und ob die Änderung online wirksam wird:

- *Konfigurierbar*

  Eine Bandbreite von Werten ist möglich, so dass der Parameter nach dem Kenntnisstand des Datenbankadministrators über die Anwendungen oder mit entsprechender Testerfahrung optimiert werden muss.

- *Online konfigurierbar*

  Eine Bandbreite von Werten ist möglich, so dass der Parameter nach dem Kenntnisstand des Datenbankadministrators über die Anwendungen und/oder mit entsprechender Testerfahrung optimiert werden muss. Änderungen können wirksam angewendet werden, während die Datenbank online ist, ohne dass der Datenbankmanager gestoppt und erneut gestartet oder die Datenbank erneut aktiviert werden muss.

- *Informativ*

  Die Werte dieser Parameter werden nur durch den Datenbankmanager selbst geändert und enthalten Informationen, wie z. B. das Release von DB2, unter dem eine Datenbank erstellt wurde, oder die Angabe, dass eine erforderliche Sicherung ansteht.

Kapazitätsverwaltung

Es gibt eine Reihe von Konfigurationsparametern sowohl auf Datenbank- als auch auch Datenbankmanagerebene, die Auswirkungen auf die Leistung des Systems haben können. Diese Parameter können in folgende Kategorien eingeteilt werden:

- „Gemeinsam benutzter Datenbankspeicher"
- „Gemeinsamer Anwendungsspeicher" auf Seite 406
- „Privater Agentenspeicher" auf Seite 410
- „Agenten-/Anwendungskommunikationsspeicher" auf Seite 421
- „Speicher des Datenbankmanagerexemplars" auf Seite 425
- „Sperren" auf Seite 431
- „Ein-/Ausgabe und Speicher" auf Seite 435
- „Agenten" auf Seite 442
- „Gespeicherte Prozeduren und benutzerdefinierte Funktionen" auf Seite 452

Gemeinsam benutzter Datenbankspeicher

Die folgenden Parameter wirken sich auf den globalen Datenbankspeicher aus, der auf dem System zugeordnet wird:

- „catalogcache_sz - Katalogcachegröße" auf Seite 396
- „database_memory - Größe des gemeinsam benutzten Datenbankspeichers" auf Seite 398
- „dbheap - Zwischenspeicher für Datenbank" auf Seite 399

- „locklist - Maximaler Speicher für Sperrenliste" auf Seite 399
- „logbufsz - Protokollpuffergröße" auf Seite 402
- „pckcachesz - Größe des Paketcache" auf Seite 403
- „sheapthres_shr - Schwellenwert für Sortierspeicher für gemeinsame Sortiervorgänge" auf Seite 405
- „util_heap_sz - Zwischenspeichergröße für Dienstprogramme" auf Seite 406

catalogcache_sz - Katalogcachegröße

Konfigurationstyp	Datenbank
Parametertyp	Online konfigurierbar
Weitergabeklasse	Sofort
Standardwert [Bereich]	-1 [8 – 524 288]
Maßeinheit	Seiten (4 KB)
Zuordnung	Wenn die Datenbank initialisiert wird
Freigabe	Wenn die Datenbank heruntergefahren wird

Dieser Parameter wird aus dem gemeinsam benutzten Datenbankspeicher zugeordnet und für das Caching (Zwischenspeichern) von Systemkataloginformationen verwendet. In einem partitionierten Datenbanksystem gibt es für jede Datenbankpartition einen Katalogcache.

Durch das Caching von Kataloginformationen in einzelnen Partitionen kann der Datenbankmanager den internen Systemaufwand verringern, da der Zugriff auf die Systemkataloge (oder auf den Katalogknoten in einer Umgebung mit partitionierten Datenbanken) nicht mehr erforderlich ist, um zuvor abgerufene Informationen zu erhalten. Der Katalogcache wird zum Speichern folgender Informationen verwendet:

- SYSTABLES-Informationen (einschließlich gepackter Deskriptoren)
- Berechtigungsinformationen, einschließlich SYSDBAUTH-Informationen und Zugriffsrechte für das Ausführen von Routinen
- SYSROUTINES-Informationen

Das Verwenden des Katalogcache kann helfen, die Gesamtleistung folgender Operationen und Anwendungen zu erhöhen:

- Binden von Paketen und Kompilieren von SQL-Anweisungen
- Operationen, die eine Überprüfung der Zugriffsrechte auf Datenbankebene umfassen
- Operationen, die eine Überprüfung der Zugriffsrechte für das Ausführen von Routinen umfassen
- Anwendungen, die mit Nicht-Katalogknoten in einer Umgebung mit partitionierten Datenbanken verbunden sind

Durch Definieren des Standardwerts (-1) in einer Serverumgebung oder in einer Umgebung mit partitionierten Datenbanken wird als Wert für die Berechnung der Seitenzuordnung das Vierfache des Werts für den Konfigurationsparameter *maxappls* genommen. Wenn das Vierfache 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 mit Hilfe 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-Kompilierungen umfasst und anschließend nur noch wenige oder gar keine SQL-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.

Mit Hilfe 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 ausdrücklich 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 Arbeitseinheit mehrere dynamische SQL-Anweisungen enthält oder wenn Sie Pakete binden, die eine große Anzahl statischer SQL-Anweisungen enthalten.

Zugehörige Referenzen:

- „maxappls - Maximale Anzahl aktiver Anwendungen" auf Seite 447
- „cat_cache_lookups - Catalog Cache Lookups monitor element" in *System Monitor Guide and Reference*
- „cat_cache_inserts - Catalog Cache Inserts monitor element" in *System Monitor Guide and Reference*
- „cat_cache_overflows - Catalog Cache Overflows monitor element" in *System Monitor Guide and Reference*
- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*
- „cat_cache_size_top - Catalog Cache High Water Mark monitor element" in *System Monitor Guide and Reference*

database_memory - Größe des gemeinsam benutzten Datenbankspeichers

Konfigurationstyp	Datenbank
Gilt für	• Datenbankserver mit lokalen und fernen Clients • Datenbankserver mit lokalen Clients • Partitionierten Datenbankserver mit lokalen und fernen Clients
Parametertyp	Konfigurierbar
Standardwert [Bereich]	Automatic [0 — 4 294 967 295]
Maßeinheit	Seiten (4 KB)
Zuordnung	Wenn die Datenbank aktiviert ist
Freigabe	Wenn die Datenbank inaktiviert ist

Dieser Parameter gibt die Menge des gemeinsam benutzten Speichers an, die für den gemeinsam benutzten Speicherbereich einer Datenbank reserviert ist. Ist diese Menge geringer als die aus den einzelnen Parametern errechnete Menge (beispielsweise aus Sperrenlisten, Zwischenspeichern von Dienstprogrammen, Pufferpools etc.) wird die größere Menge verwendet.

Um die Verwaltung dieses Parameters zu vereinfachen, wird DB2 durch die Einstellung AUTOMATIC angewiesen, die benötigte Speichermenge zu berechnen und den Speicher bei der Aktivierung der Datenbank zuzuordnen. DB2 ordnet zudem einigen zusätzlichen Speicher für einen Überlaufpuffer zu. Der Überlaufpuffer dient zur Erfüllung des Spitzenspeicherbedarfs für einen Zwischenspeicher im Bereich des gemeinsamen Datenbankspeichers, wenn ein Zwischenspeicher seine konfigurierte Größe überschreitet. Andere Operationen, wie zum Beispiel dynamische Aktualisierungen der Konfiguration, haben ebenfalls Zugriff auf diesen Überlaufpuffer. Der Befehl **db2pd** mit der Option -memsets kann zur Überwachung des im Überlaufpuffer verbliebenen nicht genutzten Speichers verwendet werden. Unter 64-Bit-DB2 für AIX wächst die Nutzung des realen gemeinsam benutzen Speichers der Datenbank bis zu einer Größe von 64 GB dynamisch an, um den Anforderungen der Datenbank Rechnung zu tragen, so dass der Parameter DATABASE_MEMORY nicht explizit gesteuert werden muss. Die Einstellung der Registriervariablen DB2_PINNED_BP oder DB2_LGPAGE_BP schränkt jedoch die Möglichkeit zur Vergrößerung des gemeinsamen Datenbankspeichers ein. Eine Beschreibung dieser Registriervariablen finden Sie unter „Leistungsvariablen".

Empfehlung: Dieser Wert bleibt normalerweise auf AUTOMATIC. Er kann jedoch auch zum Reservieren von zusätzlichem Speicher für eine zukünftige Erweiterung verwendet werden. Mit dem zusätzlichen Speicher können beispielsweise neue Pufferpools erstellt werden, oder es kann die Größe vorhandener Pufferpools erhöht werden.

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*
- „Leistungsvariablen" auf Seite 586
- „db2pd - Monitor and Troubleshoot DB2 Command" in *Command Reference*

dbheap - Zwischenspeicher für Datenbank

Konfigurationstyp	Datenbank
Parametertyp	Online konfigurierbar
Weitergabeklasse	Sofort
Standardwert [Bereich]	
	UNIX 1200 [32 – 524 288]
	Windows-Datenbankserver mit lokalen und fernen Clients 600 [32 – 524 288]
	Windows-64-Bit-Datenbankserver mit lokalen Clients 600 [32 – 524 288]
	Windows-32-Bit-Datenbankserver mit lokalen Clients 300 [32 – 524 288]
Maßeinheit	Seiten (4 KB)
Zuordnung	Wenn die Datenbank aktiviert ist
Freigabe	Wenn die Datenbank inaktiviert ist

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 der gesamte Überlaufspeicherbereich im gemeinsamen Datenbankspeicher belegt ist.

Mit Hilfe des Datenbanksystemmonitors können Sie unter Verwendung des Elements *db_heap_top* (Maximal zugeordneter Datenbankzwischenspeicher) die größte Speichermenge ermitteln, die für den Datenbankzwischenspeicher verwendet wurde.

Zugehörige Referenzen:

- „logbufsz - Protokollpuffergröße" auf Seite 402
- „db_heap_top - Maximum Database Heap Allocated monitor element" in *System Monitor Guide and Reference*
- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

locklist - Maximaler Speicher für Sperrenliste

Konfigurationstyp	Datenbank
Parametertyp	Online konfigurierbar
Weitergabeklasse	Sofort

Standardwert [Bereich]

UNIX	100 [4 – 524 288]
Windows-Datenbankserver mit lokalen und fernen Clients	50 [4 – 524 288]
Windows-64-Bit-Datenbankserver mit lokalen Clients	50 [4 – 60 000]
Windows-32-Bit-Datenbankserver mit lokalen Clients	25 [4 – 60 000]

Maßeinheit	Seiten (4 KB)
Zuordnung	Wenn die erste Anwendung die Verbindung zur Datenbank herstellt
Freigabe	Wenn die letzte Anwendung die Verbindung zur Datenbank beendet

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. 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.

Dieser Parameter kann online geändert werden, sein Wert kann dabei jedoch nur erhöht, nicht verringert werden. Wenn Sie den Wert des Parameters *locklist* verringern wollen, müssen Sie die Datenbank erneut aktivieren.

Auf 32-Bit-Plattformen erfordert jede Sperre 36 oder 72 Byte der Sperrenliste, je nachdem, ob für das Objekt andere Sperren aktiv sind:

- 72 Byte sind erforderlich, um eine Sperre für ein Objekt zu aktivieren, das keine anderen Sperren hat.
- 36 Byte sind erforderlich, um eine Sperre für ein Objekt einzutragen, für das bereits eine vorhandene Sperre aktiv ist.

Auf 64-Bit-Plattformen erfordert jede Sperre 56 oder 112 Byte der Sperrenliste, je nachdem, ob für das Objekt andere Sperren aktiv sind:

- 112 Byte sind erforderlich, um eine Sperre für ein Objekt zu aktivieren, für das keine anderen Sperren aktiviert sind.
- 56 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 (siehe unten). 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 Festschreibungen (COMMIT-Operationen) 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.

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 gegenseitige Sperren zwischen Anwendungen geben (da sie alle auf eine verringerte Anzahl von Tabellensperren warten), was dazu führt, dass Transaktionen 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 * x * maxappls) / 4096`
   b. Bei aktiviertem Konzentrator:

      `(512 * x * max_coordagents) / 4096`
   c. In einer partitionierten Datenbank mit aktiviertem Konzentrator:

      `(512 * x * max_coordagents * Anzahl Datenbankpartitionen) / 4096`

   In diesen Formeln ist 512 ein Schätzwert für die durchschnittliche Anzahl von Sperren pro Anwendung, und `x` ist die Anzahl der benötigten Byte für jede Sperre eines Objekts, für das bereits eine Sperre aktiv ist (36 Byte auf 32-Bit-Plattformen, 56 Byte auf 64-Bit-Plattformen).
2. Berechnen Sie eine Obergrenze für die Größe der Sperrenliste:

   `(512 * y * maxappls) / 4096`

   In dieser Formel ist `y` die Anzahl der benötigten Byte für die erste Sperre eines Objekts (72 Byte auf 32-Bit-Plattformen, 112 Byte auf 64-Bit-Plattformen).
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 unten beschrieben, den Wert dieses Parameters optimieren.

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 aktiver 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 Festschreibungen (COMMIT-Operationen) ausführen.

Wenn Sie diesen Parameter geändert haben, sollten Sie Anwendungen eventuell erneut binden (mit dem Befehl REBIND).

Zugehörige Referenzen:

- „maxlocks - Maximale Anzahl Sperren pro Anwendung" auf Seite 433
- „maxappls - Maximale Anzahl aktiver Anwendungen" auf Seite 447
- „lock_escals - Number of Lock Escalations monitor element" in *System Monitor Guide and Reference*
- „locks_held_top - Maximum Number of Locks Held monitor element" in *System Monitor Guide and Reference*
- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*
- „REBIND Command" in *Command Reference*

logbufsz - Protokollpuffergröße

Konfigurationstyp Datenbank

Parametertyp Konfigurierbar

Standardwert [Bereich]

32-Bit-Plattformen
8 [4 — 4 096]

64-Bit-Plattformen
8 [4 — 65 535]

Maßeinheit Seiten (4 KB)

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. Die Protokollsätze werden bei Eintreten eines der folgenden Umstände 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 viel Speicherbereich des Protokollpuffers für eine bestimmte Transaktion (oder Arbeitseinheit) verwendet wird. Weitere Informationen finden Sie in der Beschreibung des Monitorelements *log_space_used* (verwendeter Speicherbereich für Arbeitseinheit).

Zugehörige Referenzen:

- „mincommit - Anzahl der Gruppenfestschreibungen" auf Seite 471
- „dbheap - Zwischenspeicher für Datenbank" auf Seite 399
- „uow_log_space_used - Unit of Work Log Space Used monitor element" in *System Monitor Guide and Reference*
- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

pckcachesz - Größe des Paketcache

Konfigurationstyp	Datenbank
Parametertyp	Online konfigurierbar
Weitergabeklasse	Sofort
Standardwert [Bereich]	
	32-Bit-Plattformen -1 [-1, 32 — 128 000]
	64-Bit-Plattformen -1 [-1, 32 — 524 288]
Maßeinheit	Seiten (4 KB)
Zuordnung	Wenn die Datenbank initialisiert wird
Freigabe	Wenn die Datenbank heruntergefahren wird

Dieser Parameter wird aus dem gemeinsam benutzten Datenbankspeicher zugeordnet und für das Caching (Zwischenspeichern) von Abschnitten statischer und dynamischer SQL-Anweisungen in einer Datenbank verwendet. In einem partitionierten Datenbanksystem gibt es für jede Datenbankpartition einen Paketcache.

Das Caching von Paketen ermöglicht dem Datenbankmanager, den internen Systemaufwand zu verringern, da der Zugriff auf die Systemkataloge beim erneu-

ten Laden eines Pakets oder bei dynamischem SQL eine Kompilierung nicht mehr erforderlich ist. Die Abschnitte werden im Paketcache behalten, bis einer der folgenden Umstände eintritt:

- Die Datenbank wird heruntergefahren.
- Das Paket oder die dynamische SQL-Anweisung wird ungültig gemacht.
- Im Cache ist nicht mehr genügend Platz.

Dieses Zwischenspeichern des Abschnitts für eine statische oder dynamische SQL-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 Anwendung wichtig, die Transaktionen verarbeitet.

Durch Definieren des Standardwerts (-1) wird als Wert für die Berechnung der Seitenzuordnung das Achtfache des Werts für den Konfigurationsparameter *maxappls* genommen. Wenn das Achtfache von *maxappls* jedoch kleiner als 32 ist, gilt dies nicht. In diesem Fall entspricht der Standardwert -1 für *pckcachesz* dem Wert 32.

Empfehlung: Beim Optimieren dieses Parameters sollten Sie überlegen, ob der zusätzliche Speicher, der für den Paketcache reserviert wird, günstiger für einen anderen Zweck zugeordnet werden sollte, 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.

Mit Hilfe 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-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 Monitor-

element *pkg_cache_num_overflows* können Sie ermitteln, wie häufig der durch den Parameter *pckcachesz* angegebene Grenzwert überschritten wurde.

Zugehörige Referenzen:

- „maxappls - Maximale Anzahl aktiver Anwendungen" auf Seite 447
- „pkg_cache_lookups - Package Cache Lookups monitor element" in *System Monitor Guide and Reference*
- „pkg_cache_inserts - Package Cache Inserts monitor element" in *System Monitor Guide and Reference*
- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*
- „pkg_cache_num_overflows - Package Cache Overflows monitor element" in *System Monitor Guide and Reference*
- „pkg_cache_size_top - Package Cache High Water Mark monitor element" in *System Monitor Guide and Reference*

sheapthres_shr - Schwellenwert für Sortierspeicher für gemeinsame Sortiervorgänge

Konfigurationstyp Datenbank

Parametertyp Konfigurierbar

Standardwert [Bereich]

32-Bit-Plattformen
sheapthres [250 — 2 097 152]

64-Bit-Plattformen
sheapthres [250 — 2 147 483 647]

Maßeinheit Seiten (4 KB)

Dieser Parameter gibt einen festen Grenzwert für die Gesamtmenge des gemeinsam benutzten Datenbankspeichers an, die gleichzeitig von Sortiervorgängen verwendet werden kann. Wenn die Gesamtmenge des gemeinsam benutzten Speichers für aktive gemeinsame Sortiervorgänge diesen Grenzwert erreicht, schlagen nachfolgende Sortiervorgänge fehl (SQL0955C). Wenn der Parameter *sheapthres_shr* den Wert 0 hat, ist der Schwellenwert für gemeinsam benutzten Sortierspeicher gleich dem Wert des Konfigurationsparameters *sheapthres* des Datenbankmanagers, mit dem auch der Sortierspeicherschwellenwert für private Sortiervorgänge angegeben wird. Ist der Wert für *sheapthres_shr* ungleich Null, wird dieser Wert ungleich Null als Schwellenwert für den gemeinsam benutzten Sortierspeicher verwendet.

sheapthres_shr ist nur in zwei Fällen von Bedeutung:

- 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 Sortiervorgänge, die einen mit der Option WITH HOLD deklarierten Cursor verwenden, vom gemeinsam benutzten Speicher zugeordnet werden.

Zugehörige Referenzen:

- „sheapthres - Schwellenwert für Sortierspeicher" auf Seite 416
- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

util_heap_sz - Zwischenspeichergröße für Dienstprogramme

Konfigurationstyp	Datenbank
Parametertyp	Online konfigurierbar
Weitergabeklasse	Sofort
Standardwert [Bereich]	5000 [16 – 524 288]
Maßeinheit	Seiten (4 KB)
Zuordnung	Wenn für die Dienstprogramme des Datenbankmanagers erforderlich
Freigabe	Wenn der Speicher nicht mehr vom Dienstprogramm benötigt wird

Dieser Parameter gibt die maximale Speichergröße an, die gleichzeitig von den Dienstprogrammen BACKUP, RESTORE und LOAD (einschließlich Wiederherstellung) verwendet werden kann.

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 wurde und kein Speicher im Überlaufbereich mehr verfügbar sind, können Sie die Dienstprogramme eventuell nicht mehr 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 großen Wert setzen.

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

Gemeinsamer Anwendungsspeicher

Die folgenden Parameter geben den Arbeitsbereich an, der von allen Agenten (sowohl koordinierende als auch Subagenten) verwendet wird, die für eine Anwendung arbeiten:

- „app_ctl_heap_sz - Zwischenspeichergröße für Anwendungssteuerung"
- „appgroup_mem_sz - Maximale Speichergröße für Anwendungsgruppe" auf Seite 409
- „groupheap_ratio - Für Anwendungsgruppenzwischenspeicher vorgesehener Speicher in Prozent" auf Seite 410

app_ctl_heap_sz - Zwischenspeichergröße für Anwendungssteuerung

Konfigurationstyp	Datenbank

Parametertyp	Konfigurierbar
Standardwert [Bereich]	
	Datenbankserver mit lokalen und fernen Clients 128 [1 – 64 000]
	Datenbankserver mit lokalen Clients 64 [1–64 000] (Nicht-UNIX-Plattformen)
	128 [1–64 000] (UNIX-Plattformen)
	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

Bei partitionierten und nicht partitionierten Datenbanken mit aktivierter partitionsinterner Parallelität (intra_parallel=ON) gibt dieser Parameter die durchschnittliche Größe des gemeinsam benutzten 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 Partition gibt es einen Zwischenspeicher für Anwendungssteuerung.

Der Zwischenspeicher für Anwendungssteuerung wird in erster Linie für die gemeinsame Benutzung 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.

Zugehörige Referenzen:

- „intra_parallel - Partitionsinterne Parallelität aktivieren" auf Seite 522
- „applheapsz - Zwischenspeichergröße für Anwendungen" auf Seite 412
- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*
- „appgroup_mem_sz - Maximale Speichergröße für Anwendungsgruppe" auf Seite 409
- „groupheap_ratio - Für Anwendungsgruppenzwischenspeicher vorgesehener Speicher in Prozent" auf Seite 410

appgroup_mem_sz - Maximale Speichergröße für Anwendungsgruppe

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)

Dieser Parameter legt die Größe des von der Anwendungsgruppe gemeinsam benutzten Speichersegments fest. Informationen, die den für dieselbe Anwendung arbeitenden Agenten gemeinsam zur Verfügung stehen müssen, werden im gemeinsam benutzten Speichersegment der Anwendungsgruppe gespeichert.

In einer partitionierten Datenbank oder in einer nicht partitionierten Datenbank mit aktivierter partitionsinterner Parallelität oder aktiviertem Konzentrator benutzen mehrere Anwendungen eine Anwendungsgruppe gemeinsam. Der Anwendungsgruppe wird ein gemeinsam benutztes Speichersegment zugeordnet. Innerhalb des gemeinsam benutzten Speichersegments der Anwendungsgruppe ist für jede Anwendung ein eigener Zwischenspeicher für die Anwendungssteuerung vorhanden, und alle Anwendungen der Anwendungsgruppe benutzen 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 benutzten 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.

Zugehörige Referenzen:

- „app_ctl_heap_sz - Zwischenspeichergröße für Anwendungssteuerung" auf Seite 406
- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*
- „groupheap_ratio - Für Anwendungsgruppenzwischenspeicher vorgesehener Speicher in Prozent" auf Seite 410

groupheap_ratio - Für Anwendungsgruppenzwischenspeicher vorgesehener Speicher in Prozent

Konfigurationstyp	Datenbank
Parametertyp	Konfigurierbar
Standardwert [Bereich]	70 [1 – 99]
Maßeinheit	Prozent

Dieser Parameter gibt den Prozentsatz an Speicher im gemeinsam benutzten Speicher zur Anwendungssteuerung an, der für den von der Anwendungsgruppe gemeinsam benutzten Zwischenspeicher vorgesehen ist.

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.

Zugehörige Referenzen:

- „app_ctl_heap_sz - Zwischenspeichergröße für Anwendungssteuerung" auf Seite 406
- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*
- „appgroup_mem_sz - Maximale Speichergröße für Anwendungsgruppe" auf Seite 409

Privater Agentenspeicher

Die folgenden Parameter wirken sich auf die Menge an Speicher aus, die für jeden Datenbankagenten verwendet wird:

- „agent_stack_sz - Größe des Agentenstacks" auf Seite 411
- „applheapsz - Zwischenspeichergröße für Anwendungen" auf Seite 412
- „min_priv_mem - Mindestgröße des reservierten privaten Speichers" auf Seite 413
- „priv_mem_thresh - Schwellenwert für privaten Speicher" auf Seite 414

- „query_heap_sz - Größe des Abfragezwischenspeichers" auf Seite 415
- „sheapthres - Schwellenwert für Sortierspeicher" auf Seite 416
- „sortheap - Zwischenspeichergröße für Sortierlisten" auf Seite 418
- „stat_heap_sz - Größe des Statistikzwischenspeichers" auf Seite 419
- „stmtheap - Größe des Anweisungszwischenspeichers" auf Seite 420

agent_stack_sz - Größe des Agentenstacks

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]	16 [8 – 1000]
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

Der Agentenstapelspeicher ist der virtuelle Speicherbereich, der von DB2 für jeden Agenten zugeordnet wird. Dieser Speicher wird zugeordnet, wenn er zur Verarbeitung einer SQL-Anweisung angefordert wird. 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 Stapelspeicherbereich.

Mit diesem Parameter wird die ursprüngliche 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. Wenn Sie jedoch eine große SQL-Anweisung vorbereiten, kann möglicherweise ein größerer Stackbereich erforderlich sein, und das System generiert eine Ausnahmebedingung aufgrund eines Stacküberlaufs (0xC000000D). Wenn dies eintritt, fährt der Server herunter, da der Fehler nicht behebbar ist.

Die Größe des Agentenstacks kann erhöht werden, indem Sie den Parameter *agent_stack_sz* auf einen höheren Wert als die Standardreservestackgröße von 64 Seiten setzen. 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 mit Hilfe 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 erneut starten, damit die an `db2syscs.exe` vorgenommenen Änderungen wirksam werden.

Empfehlung: In den meisten Fällen kann die Standardstackgröße verwendet werden. Sie sollten den Wert für diesen Parameter nur erhöhen, wenn in Ihrer Umgebung viele extrem komplexe Abfragen ausgeführt werden.

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).

Die Größe des Agentenstapelspeichers und die Anzahl gleichzeitig ausgeführter Clients stehen in einem umgekehrten Verhältnis zueinander: Durch eine Vergrößerung des Stapelspeichers wird die mögliche Anzahl der Clients verringert, die gleichzeitig ausgeführt werden können. Die Ursache hierfür ist, dass der Adressraum auf Windows-Plattformen begrenzt ist.

Dieser Parameter gilt nicht für auf UNIX basierende Plattformen.

Zugehörige Referenzen:

- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

applheapsz - Zwischenspeichergröße für Anwendungen

Konfigurationstyp Datenbank

Parametertyp Konfigurierbar

Standardwert [Bereich]

32-Bit-Datenbankserver mit lokalen und fernen Clients
256 [16 – 60 000]

64-Bit-Datenbankserver mit lokalen und fernen Clients
256 [16 – 60 000]

Partitionierter 32-Bit-Datenbankserver mit lokalen und fernen Clients
64 [16 – 60 000]

Partitionierter 64-Bit-Datenbankserver mit lokalen und fernen Clients
128 [16 – 60 000]

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

Mit diesem Parameter wird die Anzahl der Seiten an privatem Speicher definiert, die zur Verwendung durch den Datenbankmanager für einen bestimmten Agenten oder Subagenten zur Verfügung stehen.

Der Zwischenspeicher wird zugeordnet, wenn ein Agent oder Subagent für eine Anwendung initialisiert wird. Die zugeordnete Speichermenge ist die Mindestmenge, die zur Verarbeitung der an den Agenten oder Subagenten übergebenen Anforderung benötigt wird. Wenn der Agent oder Subagent mehr Zwischenspeicher zur Verarbeitung größerer SQL-Anweisungen benötigt, ordnet der Datenbankmanager weiteren Speicher nach Bedarf bis zu der durch diesen Parameter angegebenen Obergrenze zu.

Anmerkung: In einer Umgebung mit partitionierten Datenbanken wird der Zwischenspeicher für die Anwendungssteuerung (*app_ctl_heap_sz*) verwendet, um Kopien der in Ausführung befindlichen Abschnitte von SQL-Anweisungen für Agenten und Subagenten zu speichern. SMP-Subagenten verwenden hingegen den Parameter *applheapsz*, wie es auch Agenten in allen anderen Umgebungen tun.

Empfehlung: Erhöhen Sie den Wert dieses Parameters, wenn Ihre Anwendungen einen Fehler empfangen, weil im Anwendungszwischenspeicher nicht genügend Speicher verfügbar ist.

Der Anwendungszwischenspeicher (*applheapsz*) wird aus dem privaten Agentenspeicher heraus zugeordnet.

Zugehörige Referenzen:

- „app_ctl_heap_sz - Zwischenspeichergröße für Anwendungssteuerung" auf Seite 406
- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

min_priv_mem - Mindestgröße des reservierten privaten Speichers

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]	32 [32 – 112 000]
Maßeinheit	Seiten (4 KB)
Zuordnung	Wenn der Datenbankmanager gestartet wird
Freigabe	Wenn der Datenbankmanager gestoppt wird

Mit diesem Parameter wird die Anzahl der Seiten angegeben, die der Datenbankserverprozess als privaten virtuellen Speicher reserviert, wenn ein Datenbankmanagerexemplar gestartet wird (`db2start`). Wenn der Server mehr privaten Speicher benötigt, versucht er gegebenenfalls, mehr Speicher vom Betriebssystem zu erhalten.

Dieser Parameter gilt nicht für auf UNIX basierende Systeme.

Empfehlung: Verwenden Sie den Standardwert.

Sie sollten den Wert dieses Parameters nur ändern, wenn Sie mehr Speicher für den Datenbank-Server reservieren möchten. Dadurch wird der Zeitaufwand für die Zuordnung verringert. Allerdings sollte dieser Wert nicht zu hoch gesetzt werden, da die Leistung anderer Nicht-DB2-Anwendungen beeinträchtigt werden kann.

Zugehörige Referenzen:

- „priv_mem_thresh - Schwellenwert für privaten Speicher" auf Seite 414
- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

priv_mem_thresh - Schwellenwert für privaten Speicher

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 auf UNIX basierende 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 sicher, dass Speicher für die Clientprozesse verfügbar ist, und verringert so den Systemaufwand, der durch das Zuordnen und Freigeben von Speicher entsteht.

Zugehörige Referenzen:

- „min_priv_mem - Mindestgröße des reservierten privaten Speichers" auf Seite 413
- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

query_heap_sz - Größe des Abfragezwischenspeichers

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]	1000 [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 zum Exemplar trennt

Mit diesem Parameter wird die **maximale** Speichermenge angegeben, die für den Abfragezwischenspeicher zugeordnet werden kann. 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. Mit diesem Parameter kann sichergestellt werden, dass eine Anwendung nicht unnötig große virtuelle Speicherbereiche innerhalb eines Agenten belegt.

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.

Zugehörige Referenzen:

- „aslheapsz - Zwischenspeichergröße für Anwendungsunterstützungsebene" auf Seite 421
- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

sheapthres - Schwellenwert für Sortierspeicher

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]

UNIX-Plattformen (32 Bit)
20 000 [250 — 2 097 152]
Windows-Plattformen
10 000 [250 — 2 097 152]
64-Bit-Plattformen
20 000 [250 — 2 147 483 647]

Maßeinheit Seiten (4 KB)

Private und gemeinsame Sortiervorgänge verwenden Speicher aus verschiedenen Speicherquellen. Die Größe des gemeinsamen Sortierspeicherbereichs wird auf der Grundlage des Werts von *sheapthres* statisch vorherbestimmt, wenn die erste Verbindung zu einer Datenbank hergestellt wird. Die Größe des privaten Sortierspeicherbereichs ist nicht begrenzt.

Der Parameter *sheapthres* wird für private und gemeinsame Sortiervorgänge unterschiedlich verwendet:

- Bei privaten Sortiervorgängen ist dieser Parameter ein exemplarweiter *veränderlicher Grenzwert* für die maximale Speichermenge, die zu einem beliebigen Zeitpunkt von privaten Sortiervorgängen beansprucht werden kann. Wenn der gesamte private Sortierspeicherbereich für ein Exemplar diesen Grenzwert erreicht, wird der für zusätzliche eingehende private Sortieranforderungen zugeordnete Speicherbereich beträchtlich verkleinert.
- Bei gemeinsamen Sortiervorgängen ist dieser Parameter ein datenbankweiter fester Grenzwert für den gesamten Speicherbereich, der zu einem beliebigen Zeitpunkt von gemeinsamen Sortiervorgängen belegt werden kann. Wenn dieser Grenzwert erreicht wird, sind keine weiteren Speicheranforderungen für gemeinsame Sortiervorgänge mehr möglich (bis der Speicherbereich, der für gemeinsame Sortiervorgänge belegt ist, wieder unter den von *sheapthres* angegebenen Grenzwert fällt). (Eine alternative Methode, den Maximalwert für den gemeinsamen Sortierspeicher zu konfigurieren, ist unter bestimmten Umständen die Verwendung des Datenbankkonfigurationsparameters *sheapthres_shr*.)

Der Sortierspeicher wird beispielsweise bei folgenden Operationen verwendet: Sortierungen, Hashverknüpfungen, dynamische Bitzuordnungen (die für logisches Verknüpfen von Indizes über AND (Index ANDing) und Star Joins 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.

Der Parameter *Schwellenwert für Sortierspeicher*, ein Konfigurationsparameter des Datenbankmanagers, gilt für das gesamte DB2-Exemplar. Sie können diesen Parameter in anderen Knoten oder Partitionen nur auf unterschiedliche Werte setzen, indem Sie mehrere DB2-Exemplare erstellen. Dies erfordert das Verwalten verschiedener DB2-Datenbanken in verschiedenen Datenbankpartitionsgruppen. Ein solches Vorgehen macht viele Vorteile einer Umgebung mit partitionierten Datenbanken zunichte.

Empfehlung: Im Idealfall sollten Sie den Wert dieses Parameters möglichst auf ein angemessenes Vielfaches des Parameters *sortheap* mit dem größten Wert setzen, der in Ihrem Datenbankmanagerexemplar definiert ist. Der Wert dieses Parameters sollte **mindestens** zweimal so groß sein wie der größte Sortierspeicher (*sortheap*), der für eine Datenbank im Exemplar 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 des Exemplars verwendet werden kann.

Sie sollten Vergleichstests (Benchmark-Tests) ausführen, um die optimale Einstellung für diesen Parameter zu ermitteln, die Sortierleistung und Speicherbedarf gleichermaßen berücksichtigt. Mit Hilfe des Datenbanksystemmonitors können Sie unter Verwendung des Monitorelements *post_threshold_sorts* (Sortierungen nach Erreichen des Schwellenwerts) die Sortieraktivitäten verfolgen.

Zugehörige Referenzen:

- „sortheap - Zwischenspeichergröße für Sortierlisten" auf Seite 418
- „post_threshold_sorts - Post Threshold Sorts monitor element" in *System Monitor Guide and Reference*
- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „sheapthres_shr - Schwellenwert für Sortierspeicher für gemeinsame Sortiervorgänge" auf Seite 405

sortheap - Zwischenspeichergröße für Sortierlisten

Konfigurationstyp	Datenbank
Parametertyp	Online konfigurierbar
Weitergabeklasse	Anweisungsgrenzwert
Standardwert [Bereich]	
	32-Bit-Plattformen 256 [16 – 524 288]
	64-Bit-Plattformen 256 [16 – 4 194 303]
Maßeinheit	Seiten (4 KB)
Zuordnung	Wie zur Ausführung von Sortiervorgängen erforderlich
Freigabe	Wenn der Sortiervorgang abgeschlossen ist

Mit diesem Parameter wird die maximale Anzahl von Seiten des privaten Speichers definiert, die für private Sortiervorgänge verwendet werden soll, bzw. die maximale Anzahl von Seiten des gemeinsamen Speichers, die für gemeinsame Sortiervorgänge verwendet werden soll. 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 Optimierungsprogramm wird anhand der vom Optimierungsprogramm bereitgestellten Informationen ein kleinerer als der durch diesen Parameter angegebene Sortierspeicher zugeordnet.

Empfehlung: Bei der Arbeit mit dem Sortierspeicher ist Folgendes zu beachten:

- Geeignete Indizes können die Verwendung des Sortierspeichers minimieren.
- Hashverknüpfungspuffer und dynamische Bitzuordnungen (die für logisches Verknüpfen von Indizes über AND (Index ANDing) und Star Joins verwendet werden) greifen auf den Sortierspeicher zu. Erhöhen Sie den Wert dieses Parameters, wenn diese Methoden verwendet werden.
- 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 der Wert des Parameters *sheapthres* in der Konfigurationsdatei des Datenbankmanagers angepasst werden muss.
- Die Größe des Zwischenspeichers für Sortierlisten wird vom Optimierungsprogramm zur Bestimmung der Zugriffspfade verwendet. Wenn Sie diesen Parameter geändert haben, sollten Sie Anwendungen eventuell erneut binden (mit dem Befehl REBIND).

Zugehörige Referenzen:

- „sheapthres - Schwellenwert für Sortierspeicher" auf Seite 416
- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*
- „REBIND Command" in *Command Reference*
- „sheapthres_shr - Schwellenwert für Sortierspeicher für gemeinsame Sortiervorgänge" auf Seite 405

stat_heap_sz - Größe des Statistikzwischenspeichers

Konfigurationstyp	Datenbank
Parametertyp	Konfigurierbar
Standardwert [Bereich]	4384 [1096 – 524 288]
Maßeinheit	Seiten (4 KB)
Zuordnung	Wenn das Dienstprogramm RUNSTATS gestartet wird
Freigabe	Wenn das Dienstprogramm RUNSTATS beendet ist

Mit diesem Parameter wird die **maximale** Größe des Zwischenspeichers angegeben, der bei Erfassung statistischer Daten mit dem Befehl RUNSTATS verwendet wird.

Empfehlung: Der Standardwert ist geeignet, wenn keine Verteilungsstatistikdaten gesammelt werden oder wenn die Verteilungsstatistikdaten nur für relativ schmale

Tabellen gesammelt werden. Der Minimalwert ist **nicht** zu empfehlen, wenn Verteilungsstatistikdaten gesammelt werden, da nur Tabellen mit einer oder zwei Spalten in den Zwischenspeicher passen.

Dieser Parameter sollte nach der Anzahl der Spalten, für die statistische Daten erfasst werden, angepasst werden. Schmale Tabellen mit relativ wenigen Spalten erfordern weniger Speicher für die zu sammelnden Verteilungsstatistikdaten. Breite Tabellen mit vielen Spalten machen erheblich mehr Speicher erforderlich. Wenn Sie Verteilungsstatistikdaten für sehr breite Tabellen sammeln und einen großen Statistikzwischenspeicher benötigen, sollten Sie die Statistikdaten in einer Phase geringer Systemaktivität sammeln, um die Speicheranforderungen anderer Benutzer nicht einzuschränken.

Zugehörige Referenzen:

- „num_freqvalues - Anzahl der häufigsten Werte" auf Seite 506
- „num_quantiles - Anzahl der Quantile für Spalten" auf Seite 507
- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „RUNSTATS Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

stmtheap - Größe des Anweisungszwischenspeichers

Konfigurationstyp	Datenbank
Parametertyp	Online konfigurierbar
Weitergabeklasse	Anweisungsgrenzwert
Standardwert [Bereich]	2048 [128 – 65 535]
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

Der Anweisungszwischenspeicher wird als Arbeitsbereich für den SQL-Compiler während des Kompilierens einer SQL-Anweisung verwendet. Mit diesem Parameter wird die Größe dieses Arbeitsbereichs angegeben.

Dieser Bereich bleibt nicht permanent zugeordnet, sondern wird für die Verarbeitung jeder SQL-Anweisung einzeln zugeordnet und anschließend wieder freigegeben. Beachten Sie, dass dieser Arbeitsbereich bei dynamischen SQL-Anweisungen während der Ausführung Ihres Programms, bei statischen SQL-Anweisungen hingegen während des Bindens, und nicht während der Ausführung des Programms, verwendet wird.

Empfehlung: In den meisten Fällen kann der Standardwert für diesen Parameter übernommen werden. Wenn Sie sehr große SQL-Anweisungen verwenden und der Datenbankmanager bei dem Versuch, eine Anweisung zu optimieren, einen Fehler ausgibt (dass die Anweisung zu komplex ist), sollten Sie den Wert dieses Parameters in gleichmäßigen Schritten (z. B. 256 oder 1024) erhöhen, bis die Fehlersituation bereinigt ist.

Zugehörige Referenzen:

- „sortheap - Zwischenspeichergröße für Sortierlisten" auf Seite 418
- „applheapsz - Zwischenspeichergröße für Anwendungen" auf Seite 412
- „stat_heap_sz - Größe des Statistikzwischenspeichers" auf Seite 419
- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

Agenten-/Anwendungskommunikationsspeicher

Die folgenden Parameter wirken sich auf die Menge an Speicher aus, der zum Übergeben von Daten zwischen den Anwendungs- und Agentenprozessen zugeordnet wird:

- „aslheapsz - Zwischenspeichergröße für Anwendungsunterstützungsebene"
- „min_dec_div_3 - 3 Kommastellen bei Dezimaldivision" auf Seite 423
- „rqrioblk - E/A-Blockgröße für Clients" auf Seite 424

aslheapsz - Zwischenspeichergröße für Anwendungsunterstützungsebene

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]
Maßeinheit	Seiten (4 KB)
Zuordnung	Wenn der Agentenprozess des Datenbankmanagers für die lokale Anwendung gestartet wird
Freigabe	Wenn der Agentenprozess des Datenbankmanagers beendet wird

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.

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 Größe des E/A-Blocks 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 Datenbankclient keinen Bereich für einen Blockcursor aus dem privaten Speicher der Anwendung zuordnen kann, wird ein Cursor ohne Blockung geöffnet.
- Mit diesem Parameter wird die Kommunikationsgröße zwischen Agenten und db2fmp-Prozessen ermittelt. (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 gemeinsam benutzten 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 vielleicht 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 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.

Zugehörige Referenzen:

- „query_heap_sz - Größe des Abfragezwischenspeichers” auf Seite 415

- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

min_dec_div_3 - 3 Kommastellen bei Dezimaldivision

Konfigurationstyp	Datenbank
Parametertyp	Konfigurierbar
Standardwert [Bereich]	No [Yes, No]

Der Datenbankkonfigurationsparameter *min_dec_div_3* bietet die Möglichkeit, die Berechnung der Kommastellen bei der Dezimaldivision in SQL schnell zu ändern. Mögliche Werte für *min_dec_div_3* sind "Yes" und "No". Der Standardwert für *min_dec_div_3* ist "No".

Der Datenbankkonfigurationsparameter *min_dec_div_3* ändert die Anzahl der Kommastellen im Ergebnis einer Dezimalrechenoperation, die eine Division umfasst. Wenn der Wert des Parameters "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. Nachfolgend werden einige mögliche Szenarios beschrieben, 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.

- Wenn die Anzahl der Kommastellen im Ergebnis für eine der Sichtspalten geändert wird, könnte eine Sicht, die in einer Umgebung mit einer Einstellung definiert ist, mit SQLCODE -344 fehlschlagen, wenn nach Änderung des Datenbankkonfigurationsparameters auf diese Sicht verwiesen wird. Die Nachricht SQL0344N verweist auf rekursive allgemeine Tabellenausdrücke, wenn der Objektname (erstes Token) jedoch eine Sicht ist, müssen Sie die Sicht löschen und sie erneut erstellen, um diesen Fehler zu vermeiden.
- 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 REBIND-Befehl 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`

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

rqrioblk - E/A-Blockgröße für Clients

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] 32 767 [4 096 – 65 535]

Maßeinheit Byte

Zuordnung

- Wenn eine ferne Client-Anwendung eine Verbindungsanforderung für eine Server-Datenbank absetzt
- Wenn ein Blockcursor geöffnet wird, werden zusätzliche Blöcke auf dem Client geöffnet

Freigabe

- Wenn die ferne Anwendung die Verbindung zur Server-Datenbank trennt
- Wenn der Blockcursor geschlossen wird

Mit diesem Parameter wird die Größe des Puffers für die Kommunikation zwischen fernen Anwendungen und ihren Datenbankagenten auf dem Datenbank-

server festgelegt. Wenn ein Datenbankclient die Verbindung zu einer fernen Datenbank anfordert, wird dieser Kommunikationspuffer auf dem Client zugeordnet. Auf dem Datenbankserver wird zu Anfang ein Kommunikationspuffer von 32 767 Byte zugeordnet, bis eine Verbindung hergestellt ist und der Server den Wert des Parameters *rqrioblk* auf dem Client ermitteln kann. Wenn der Server diesen Wert ermittelt hat, wird der Kommunikationspuffer auf dem Server neu zugeordnet, sofern der Client-Puffer nicht 32 767 Byte groß ist.

Außer für diesen Kommunikationspuffer wird dieser Parameter auch dazu verwendet, die Größe des E/A-Blocks auf dem Datenbankclient 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 Datenbankclient keinen Bereich für einen Blockcursor aus dem privaten Speicher der Anwendung zuordnen kann, wird ein Cursor ohne Blockung geöffnet.

Empfehlung: Bei Cursorn ohne Blockung könnte ein Grund zur Erhöhung des Werts für diesen Parameter darin bestehen, dass die Daten (z. B. LOB-Daten), die durch eine einzige SQL-Anweisung übertragen werden sollen, so umfangreich sind, dass der Standardwert nicht ausreicht.

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.

Zugehörige Referenzen:

- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

Speicher des Datenbankmanagerexemplars

Die folgenden Parameter wirken sich auf den Speicher aus, der auf Exemplarebene zugeordnet und verwendet wird:

- „audit_buf_sz - Prüfpuffergröße" auf Seite 426
- „dir_cache - Verzeichniscacheunterstützung" auf Seite 426
- „instance_memory - Exemplarspeicher" auf Seite 428
- „java_heap_sz - Maximale Zwischenspeichergröße für Java-Interpreter" auf Seite 429
- „mon_heap_sz - Zwischenspeichergröße für Datenbanksystemmonitor" auf Seite 430

audit_buf_sz - Prüfpuffergröße

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

Mit diesem Parameter wird die Größe des Puffers für die Prüfung der Datenbank angegeben.

Der Standardwert für diesen Konfigurationsparameter 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. 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 Null (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.

Zugehörige Referenzen:

- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

dir_cache - Verzeichniscacheunterstützung

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 ein Datenbankmanagerexemplar gestartet (`db2start`) wird, wird der Serververzeichniscache zugeordnet.
Freigabe	• Wenn ein Anwendungsprozess beendet wird, wird der Anwendungsverzeichniscache freigegeben. • Wenn ein Datenbankmanagerexemplar gestoppt (`db2stop`) wird, wird der Serververzeichniscache freigegeben.

Wenn der Parameter *dir_cache* auf "Yes" gesetzt wird, werden die Datenbank-, Knoten- und DCS-Verzeichnisdateien im Cache zwischengespeichert. 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 Caches 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 ein Datenbankmanagerexemplar gestartet wird (`db2start`), werden in einem Serververzeichniscache alle Verzeichnisdateien gelesen, und die Informationen werden im Servercache zwischengespeichert. Dieser Cache wird beibehalten, bis das Exemplar gestoppt (`db2stop`) wird. Wenn ein Verzeichniseintrag in diesem Cache nicht gefunden wird, werden die Verzeichnisdateien nach den Daten durchsucht. Dieser Serververzeichniscache wird während der Ausführung des Exemplars zu keinem Zeitpunkt aktualisiert.

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.

Zugehörige Referenzen:

- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

instance_memory - Exemplarspeicher

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 [8 — 524 288]
Maßeinheit	Seiten (4 KB)
Zuordnung	Wenn das Exemplar gestartet wird
Freigabe	Wenn das Exemplar gestoppt wird

Dieser Parameter gibt die Speicherkapazität an, die zur Exemplarverwaltung reserviert werden soll. Dies umfasst Speicherbereiche, die die Datenbanken auf dem Exemplar beschreiben.

Wenn Sie diesen Parameter auf AUTOMATIC setzen, berechnet DB2 die für die aktuelle Konfiguration benötigte Speicherkapazität für das Exemplar. DB2 ordnet zudem einigen zusätzlichen Speicher für einen Überlaufpuffer zu. Der Überlaufpuffer dient zur Erfüllung des Spitzenspeicherbedarfs für einen Zwischenspeicher im Bereich des gemeinsamen Exemplarspeichers, wenn ein Zwischenspeicher seine konfigurierte Größe überschreitet. Andere Operationen, wie zum Beispiel dynamische Aktualisierungen der Konfiguration, haben ebenfalls Zugriff auf diesen Über-

laufpuffer. Der Befehl **db2pd** mit der Option -memsets kann zur Überwachung des im Überlaufpuffer verbliebenen nicht genutzten Speichers verwendet werden.

Zugehörige Referenzen:

- „maxagents - Maximale Anzahl von Agenten" auf Seite 446
- „numdb - Maximale Anzahl gleichzeitig aktiver Datenbanken einschließlich Host- und iSeries-Datenbanken" auf Seite 533
- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „db2pd - Monitor and Troubleshoot DB2 Command" in *Command Reference*

java_heap_sz - Maximale Zwischenspeichergröße für Java-Interpreter

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]	512 [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

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.

Es gibt einen Zwischenspeicher für jeden DB2-Prozess (einen für jeden Agenten oder Subagenten auf UNIX-gestützten Plattformen und einen für jedes Exemplar auf anderen Plattformen). Es gibt einen Zwischenspeicher für jede abgeschirmte benutzerdefinierte Funktion und für jede abgeschirmte gespeicherte Prozedur. Es gibt einen Zwischenspeicher für jeden Agenten (ausschließlich der Subagenten) für gesicherte Routinen. 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. Nur die Agenten oder Prozesse, die in Java geschriebene benutzerdefinierte Funktionen oder gespeicherte Prozeduren ausführen, ordnen diesen Speicher zu. Bei partitionierten Datenbanksystemen wird in jeder Partition derselbe Wert verwendet.

Zugehörige Referenzen:

- „jdk_path - Installationspfad für das Software Developer's Kit für Java" auf Seite 532
- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

mon_heap_sz - Zwischenspeichergröße für Datenbanksystemmonitor

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]

UNIX 90 [0 – 60 000]

Windows-Datenbankserver mit lokalen und fernen Clients 66 [0 – 60 000]

Windows-Datenbankserver mit lokalen Clients 46 [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

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, wie z. B. Erstellen einer Momentaufnahme (Snapshot), Aktivieren eines Monitorschalters oder eines Ereignismonitors, Zurücksetzen eines Monitors o. ä.

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 verfügbare Speicherkapazität in diesem Zwischenspeicher erschöpft ist und der Überlaufpuffer keinen ungenutzten Speicher enthält, wird eine der folgenden Aktionen ausgeführt:

- 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, der mit Hilfe der Anweisung SET EVENT MONITOR dynamisch gestartet wird, fehlschlägt, wird ein Fehlercode an die Anwendung zurückgegeben.
- Wenn ein Monitorbefehl oder eine API-Unterroutine fehlschlägt, wird ein Fehlercode an die Anwendung zurückgegeben.

Zugehörige Konzepte:

- „Database system monitor memory requirements" in *System Monitor Guide and Reference*

Zugehörige Referenzen:

- „dft_monswitches - Monitorschalter des Standarddatenbanksystems" auf Seite 527
- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

Sperren

Die folgenden Parameter beeinflussen die Sperrenverwaltung in Ihrer Umgebung:

- „dlchktime - Zeitintervall für Prüfung gegenseitiger Sperren"
- „locktimeout - Zeitlimit für Sperren" auf Seite 432
- „maxlocks - Maximale Anzahl Sperren pro Anwendung" auf Seite 433

Siehe auch „locklist - Maximaler Speicher für Sperrenliste" auf Seite 399.

dlchktime - Zeitintervall für Prüfung gegenseitiger Sperren

Konfigurationstyp	Datenbank
Parametertyp	Online konfigurierbar
Weitergabeklasse	Sofort
Standardwert [Bereich]	10 000 (10 Sekunden) [1 000 – 600 000]
Maßeinheit	Millisekunden

Bei Auftreten gegenseitiger Sperren müssen zwei oder mehr Anwendungen, die auf dieselbe Datenbank zugreifen, unendlich lange 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.

Der Parameter für das Intervall zur Prüfung auf gegenseitige Sperren definiert die Häufigkeit, mit der vom Datenbankmanager alle mit der Datenbank verbundenen Anwendungen auf gegenseitige Sperren überprüft werden.

Anmerkungen:

1. In einer Umgebung mit partitionierten Datenbanken gilt dieser Parameter nur für den Katalogknoten.

2. In einer Umgebung mit partitionierten Datenbanken wird eine gegenseitige Sperre erst nach der zweiten Iteration markiert.

Empfehlung: Durch Erhöhung des Werts für diesen Parameter wird die Häufigkeit der Prüfung auf gegenseitige Sperren verringert, wodurch die Zeit, die Anwendungsprogramme auf die Aufhebung gegenseitiger Sperren warten müssen, erhöht wird.

Durch Angabe eines niedrigeren Werts für diesen Parameter wird die Häufigkeit der Prüfung auf gegenseitige Sperren erhöht, wodurch die Zeit, die Anwendungsprogramme auf die Aufhebung gegenseitiger Sperren warten müssen, verkürzt, die Zeit, die der Datenbankmanager auf die Prüfung gegenseitiger Sperren verwendet, jedoch verlängert wird. Wenn das Prüfintervall zu klein ist, kann dies zu einer Verschlechterung der Laufzeitleistung führen, weil der Datenbankmanager häufig nach gegenseitigen Sperren 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 gegenseitigen Sperren führen können.

Zugehörige Referenzen:

- „locklist - Maximaler Speicher für Sperrenliste" auf Seite 399
- „maxlocks - Maximale Anzahl Sperren pro Anwendung" auf Seite 433
- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

locktimeout - Zeitlimit für Sperren

Konfigurationstyp	Datenbank
Parametertyp	Konfigurierbar
Standardwert [Bereich]	-1 [-1; 0 – 32 767]
Maßeinheit	Sekunden

Mit diesem Parameter wird die Anzahl der Sekunden angegeben, die eine Anwendung auf eine Sperre wartet. Dies trägt zur Vermeidung globaler gegenseitiger Sperren von Anwendungen bei.

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 eine der folgenden Situationen eintritt:

- Die Sperre wird erteilt
- Eine gegenseitige Sperre tritt ein

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 mit Hilfe von Vergleichstests (Benchmark-Tests) optimieren.

Wenn Sie bei der Arbeit mit Data Links File Manager (DLFM) im Benachrichtigungsprotokoll für die Verwaltung des Exemplars von Data Links File Manager Überschreitungen der Sperrzeit feststellen, erhöhen Sie den Wert für den Parameter *locktimeout*. Erhöhen Sie gegebenenfalls auch den Wert für *locklist*.

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 Sperranforderungen 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. Mit Hilfe 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).

Zugehörige Referenzen:

- „locklist - Maximaler Speicher für Sperrenliste" auf Seite 399
- „maxlocks - Maximale Anzahl Sperren pro Anwendung" auf Seite 433
- „lock_timeouts - Number of Lock Timeouts monitor element" in *System Monitor Guide and Reference*
- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

maxlocks - Maximale Anzahl Sperren pro Anwendung

Konfigurationstyp	Datenbank	
Parametertyp	Online konfigurierbar	
Weitergabeklasse	Sofort	
Standardwert [Bereich]		
	UNIX	10 [1 – 100]
	Windows	22 [1 – 100]
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. Mit diesem Parameter wird definiert, wie viel Prozent

der Sperrenliste eine Anwendung belegen muss, damit der Datenbankmanager eine Eskalation ausführt. 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 ermittelt die zu eskalierenden Sperren, indem er die Sperrenliste für die Anwendung nach der Tabelle mit den meisten gesperrten Zeilen durchsucht. 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.

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 halten kann, bevor eine Sperreneskalation erfolgt, wird wie folgt berechnet:

`maxlocks * locklist * 4 096 / (100 * 36)` auf einem 32-Bit-System

`maxlocks * locklist * 4 096 / (100 * 56)` auf einem 64-Bit-System

Dabei ist `4 096` die Anzahl Byte auf einer Seite, `100` ist der höchste zulässige Prozentwert für *maxlocks*, `36` ist die Anzahl der Byte pro Sperre auf einem 32-Bit-System und `56` die Anzahl der Byte pro Sperre auf einem 64-Bit-System. Wenn Sie wissen, dass eine Ihrer Anwendungen 1 000 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 1 000 ist. (Wenn Sie für *maxlocks* 10 und für *locklist* 100 verwenden, ist das Ergebnis der Formel höher als die erforderlichen 1 000 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.

Mit Hilfe des Datenbanksystemmonitors können Sie diesen Konfigurationsparameter verfolgen und seinen Wert optimieren.

Zugehörige Referenzen:

- „locklist - Maximaler Speicher für Sperrenliste" auf Seite 399
- „maxappls - Maximale Anzahl aktiver Anwendungen" auf Seite 447
- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

Ein-/Ausgabe und Speicher

Die folgenden Parameter beeinflussen den Ein-/Ausgabeaufwand und den Speicherbedarf, die mit dem Betrieb der Datenbank verbunden sind:

- „chngpgs_thresh - Schwellenwert für geänderte Seiten"
- „dft_extent_sz - Standardwert für EXTENTSIZE bei Tabellenbereichen" auf Seite 436
- „dft_prefetch_sz - Standardwert für PREFETCHSIZE" auf Seite 436
- „estore_seg_sz - Segmentgröße für erweiterten Speicher" auf Seite 438
- „num_estore_segs - Segmentanzahl für erweiterten Speicher" auf Seite 438
- „num_iocleaners - Anzahl asynchroner Seitenlöschfunktionen" auf Seite 439
- „num_ioservers - Anzahl von E/A-Servern" auf Seite 440
- „numsegs - Standardanzahl von SMS-Behältern" auf Seite 441
- „seqdetect - Markierung für Sequenzerkennung" auf Seite 441

chngpgs_thresh - Schwellenwert für geänderte Seiten

Konfigurationstyp	Datenbank
Parametertyp	Konfigurierbar
Standardwert [Bereich]	60 [5 – 99]
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.

Mit diesem Parameter können Sie den Prozentsatz der geänderten Seiten angeben, bei dem die asynchronen Seitenlöschfunktionen gestartet werden, wenn sie zum gegebenen Zeitpunkt nicht aktiv sind. 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 Startauslöser.

In einer Umgebung mit reinem Lesezugriff (in der z. B. nur Abfragen ausgeführt werden) werden diese Seitenlöschfunktionen nicht verwendet.

Wenn die Registriervariable DB2_USE_ALTERNATE_PAGE_CLEANING definiert ist (d. h. die alternative Methode der Seitenbereinigung verwendet wird), hat der Parameter *chngpgs_thresh* keine Wirkung und DB2 bestimmt automatisch, wie viele geänderte 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.

Zugehörige Referenzen:

- „num_iocleaners - Anzahl asynchroner Seitenlöschfunktionen" auf Seite 439
- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

dft_extent_sz - Standardwert für EXTENTSIZE bei Tabellenbereichen

Konfigurationstyp	Datenbank
Parametertyp	Online konfigurierbar
Weitergabeklasse	Sofort
Standardwert [Bereich]	32 [2 – 256]
Maßeinheit	Seiten

Bei der Erstellung eines Tabellenbereichs kann wahlfrei `EXTENTSIZE n` angegeben werden, wobei n die Speicherbereichsgröße ist. Wenn Sie `EXTENTSIZE n` 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.

Zugehörige Konzepte:

- „Parameter EXTENTSIZE" in *Systemverwaltung: Konzept*

Zugehörige Referenzen:

- „dft_prefetch_sz - Standardwert für PREFETCHSIZE" auf Seite 436
- „CREATE TABLESPACE statement" in *SQL Reference, Volume 2*
- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

dft_prefetch_sz - Standardwert für PREFETCHSIZE

Konfigurationstyp	Datenbank	
Parametertyp	Online konfigurierbar	
Weitergabeklasse	Sofort	
Standardwert [Bereich]		
	UNIX	Automatic [0 — 32 767]
	Windows	Automatic [0 — 32 767]
Maßeinheit	Seiten	

Bei der Erstellung eines Tabellenbereichs kann optional PREFETCHSIZE *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 ein Tabellenbereich mit der Angabe AUTOMATIC DFT_PREFETCH_SZ erstellt wird, wird die Größe des Vorablesezugriffs des Tabellenbereichs automatisch ermittelt. Dies bedeutet, dass DB2 den Wert für PREFETCHSIZE des Tabellenbereichs nach der folgenden Gleichung automatisch berechnet und aktualisiert:

```
PREFETCHSIZE = (Anzahl Behälter) * (Anzahl physischer Spindeln) * EXTENTSIZE
```

Dabei gilt für die Anzahl der physischen Spindeln der Standardwert 1. Diese Anzahl kann über die DB2-Registriervariable DB2_PARALLEL_IO definiert werden. Diese Berechnung erfolgt zu folgenden Zeitpunkten:

- Beim Starten der Datenbank
- Bei der anfänglichen Erstellung eines Tabellenbereichs mit AUTOMATIC DFT_PREFETCH_SZ
- Bei einer Änderung der Anzahl von Behältern 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*).

Zugehörige Referenzen:

- „ALTER TABLESPACE statement" in *SQL Reference, Volume 2*

- „CREATE TABLESPACE statement" in *SQL Reference, Volume 2*
- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*
- „Systemumgebungsvariablen" auf Seite 569

estore_seg_sz - Segmentgröße für erweiterten Speicher

Konfigurationstyp	Datenbank
Parametertyp	Konfigurierbar
Standardwert [Bereich]	16 000 [0 – 1 048 575]
Maßeinheit	Seiten (4 KB)

Dieser Parameter gibt die Anzahl der Seiten in jedem Segment des erweiterten Speichers in der Datenbank an. Dieser Parameter wird nur verwendet, wenn Ihre Maschine über mehr adressierbaren Realspeicher als die maximal adressierbare virtuelle Speichermenge verfügt.

Empfehlung: Dieser Parameter ist nur wirksam, wenn erweiterter Speicher verfügbar ist, und er wird wie für den Parameter *num_estore_segs* dargestellt verwendet. Beim Angeben der in jedem Segment des erweiterten Speichers zu verwendenden Anzahl von Seiten sollten Sie auch die Segmentanzahl im erweiterten Speicher berücksichtigen, indem Sie den Parameter *num_estore_segs* überprüfen und ggf. ändern.

Zugehörige Referenzen:

- „num_estore_segs - Segmentanzahl für erweiterten Speicher" auf Seite 438
- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

num_estore_segs - Segmentanzahl für erweiterten Speicher

Konfigurationstyp	Datenbank
Parametertyp	Konfigurierbar
Standardwert [Bereich]	0 [0 – 2 147 483 647]

Dieser Parameter gibt die Anzahl der Segmente im erweiterten Speicher an, die von der Datenbank verwendet werden können.

Die Standardeinstellung ist 0, d. h., für die Datenbank werden keine Segmente im erweiterten Speicher bereitgestellt.

Empfehlung: Verwenden Sie diesen Parameter nur, um die Verwendung von Segmenten im erweiterten Speicher einzurichten, wenn der Speicherplatz Ihrer Systemumgebung den maximalen Adressraum übersteigt und Sie diesen Speicher nutzen möchten. Beim Angeben der Segmentanzahl sollten Sie auch die Größe der einzelnen Segmente berücksichtigen, indem Sie den Parameter *estore_seg_sz* überprüfen und ggf. ändern.

Wenn die Konfigurationsparameter *num_estore_segs* und *estore_seg_sz* gesetzt sind, sollten Sie mit Hilfe der Anweisungen CREATE/ALTER BUFFERPOOL angeben, welche Pufferpools den erweiterten Speicher verwenden werden.

Zugehörige Referenzen:

- „estore_seg_sz - Segmentgröße für erweiterten Speicher" auf Seite 438
- „ALTER BUFFERPOOL statement" in *SQL Reference, Volume 2*
- „CREATE BUFFERPOOL statement" in *SQL Reference, Volume 2*
- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

num_iocleaners - Anzahl asynchroner Seitenlöschfunktionen

Konfigurationstyp	Datenbank
Parametertyp	Konfigurierbar
Standardwert [Bereich]	1 [0 – 255]
Maßeinheit	Zähler

Mit diesem Parameter können Sie die Anzahl asynchroner Seitenlöschfunktionen für eine Datenbank angeben. 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.

Wenn der Parameter auf den Wert 0 gesetzt wird, werden keine Seitenlöschfunktionen gestartet, und infolgedessen schreiben die Agenten alle geänderten Seiten aus dem Pufferpool auf Platte. Dieser Parameter kann erhebliche Auswirkungen auf die Leistung einer Datenbank haben, die auf mehrere physische Speichereinheiten verteilt ist, weil in diesem Fall die Wahrscheinlichkeit größer ist, dass auf einer dieser Einheiten momentan keine Operationen ausgeführt werden. Wenn keine Seitenlöschfunktionen konfiguriert sind, könnten Ihre Anwendungen in regelmäßigen Abständen auf volle Protokolle stoßen.

Wenn die Anwendungen für eine Datenbank im Wesentlichen aus Transaktionen bestehen, mit denen die Daten aktualisiert werden, führt eine Erhöhung der Anzahl der Seitenlöschfunktionen zu einer Leistungsverbesserung. Durch die Erhöhung der Anzahl der Seitenlöschfunktionen wird außerdem die Zeit für Wiederherstellungen nach Systemausfällen, zum Beispiel aufgrund von Netzausfall, verringert, weil der Inhalt der Datenbank auf der Platte zu einem gegebenen Zeitpunkt aktueller ist.

Empfehlung: Bei der Einstellung dieses Parameters müssen folgende Faktoren beachtet werden:

- Anwendungsart
  - Wenn es sich um eine reine Abfragedatenbank handelt, die nicht aktualisiert wird, setzen Sie diesen Parameter auf den Wert Null (0). Eine Ausnahme hiervon wäre, wenn die Abfrageauslastung dazu führt, dass viele TEMP-Tabellen erstellt werden. (Verwenden Sie das Dienstprogramm EXPLAIN, um dies festzustellen.)
  - Wenn Transaktionen für die Datenbank ausgeführt werden, setzen Sie diesen Parameter auf einen Wert zwischen 1 und der Anzahl der physischen Speichereinheiten, die für diese Datenbank verwendet werden.
- Auslastung

Umgebungen mit hohem Aufkommen an aktualisierenden Transaktionen können eventuell die Konfiguration weiterer Seitenlöschfunktionen erforderlich machen.

- Pufferpoolgrößen

  Umgebungen mit großen Pufferpools können eventuell auch die Konfiguration weiterer Seitenlöschfunktionen erforderlich machen.

Mit Hilfe 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*
- 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*.

Zugehörige Referenzen:

- „chngpgs_thresh - Schwellenwert für geänderte Seiten" auf Seite 435
- „pool_data_writes - Buffer Pool Data Writes monitor element" in *System Monitor Guide and Reference*
- „pool_index_writes - Buffer Pool Index Writes monitor element" in *System Monitor Guide and Reference*
- „pool_async_data_writes - Buffer Pool Asynchronous Data Writes monitor element" in *System Monitor Guide and Reference*
- „pool_async_index_writes - Buffer Pool Asynchronous Index Writes monitor element" in *System Monitor Guide and Reference*
- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

num_ioservers - Anzahl von E/A-Servern

Konfigurationstyp	Datenbank
Parametertyp	Konfigurierbar
Standardwert [Bereich]	3 [1 – 255]
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 werden für Datenbankagenten verwendet, um Vorablese-E/A-Operationen und asynchrone E/A-Operationen von Dienstprogrammen wie BACKUP (Sicherung) und RESTORE (Wiederherstellung) auszuführen. 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. 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, so dass diese Ein-/Ausgaben nicht der Begrenzung durch den Parameter *num_ioservers* unterliegen.

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 geringfügiger Systemaufwand verbunden ist und alle nicht benötigten E/A-Server inaktiv bleiben.

Zugehörige Referenzen:

- „dft_prefetch_sz - Standardwert für PREFETCHSIZE" auf Seite 436
- „seqdetect - Markierung für Sequenzerkennung" auf Seite 441
- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

numsegs - Standardanzahl von SMS-Behältern

Konfigurationstyp	Datenbank
Parametertyp	Informativ
Maßeinheit	Zähler

Dieser Parameter, der sich ausschließlich auf SMS-Tabellenbereiche bezieht, gibt die Anzahl der Behälter an, die innerhalb der Standardtabellenbereiche erstellt werden. Dieser Parameter zeigt den Wert an, der bei der Erstellung der Datenbank verwendet wurde, unabhängig davon, ob er in dem Befehl CREATE DATABASE explizit oder implizit angegeben wurde. Von der Anweisung CREATE TABLESPACE wird dieser Parameter **nicht** verwendet.

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*

seqdetect - Markierung für Sequenzerkennung

Konfigurationstyp	Datenbank
Parametertyp	Online konfigurierbar
Weitergabeklasse	Sofort
Standardwert [Bereich]	Yes [Yes; No]

Der Datenbankmanager kann die E/A-Operationen überwachen und, wenn Seiten sequenziell gelesen werden, das E/A-Vorablesen (I/O Prefetching) aktivieren. Diese Art des sequenziellen Vorablesens ist die *Sequenzerkennung*. Mit dem Konfigurationsparameter *seqdetect* können Sie steuern, ob der Datenbankmanager die Sequenzerkennung ausführen soll.

Wenn für diesen Parameter der Wert "No" angegeben wird, findet das Vorablesen nur dann statt, wenn der Datenbankmanager erkennt, dass dies nützlich ist, z. B. bei Tabellensortierungen, Tabellensuchen oder einem Vorablesezugriff über Listen.

Empfehlung: In den meisten Fällen sollte der Standardwert für diesen Parameter verwendet werden. Inaktivieren Sie die Sequenzerkennung nur dann, wenn andere Optimierungsversuche bisher nicht zur Behebung schwerwiegender Leistungsprobleme bei Abfragen geführt haben.

Zugehörige Referenzen:

- „dft_prefetch_sz - Standardwert für PREFETCHSIZE" auf Seite 436
- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

Agenten

Die folgenden Parameter können die Anzahl der Anwendungen beeinflussen, die gleichzeitig ausgeführt werden können, um eine optimale Leistung zu erzielen:

- „agentpri - Agentenpriorität"
- „avg_appls - Durchschnittliche Anzahl aktiver Anwendungen" auf Seite 443
- „max_connections - Maximale Anzahl von Clientverbindungen" auf Seite 444
- „max_coordagents - Maximale Anzahl koordinierender Agenten" auf Seite 445
- „maxagents - Maximale Anzahl von Agenten" auf Seite 446
- „maxappls - Maximale Anzahl aktiver Anwendungen" auf Seite 447
- „maxcagents - Maximale Anzahl gleichzeitig aktiver Agenten" auf Seite 448
- „maxfilop - Maximale Anzahl offener Datenbankdateien pro Anwendung" auf Seite 449
- „maxtotfilop - Maximale Anzahl offener Dateien" auf Seite 450
- „num_initagents - Anfangswert für die Anzahl Agenten im Pool" auf Seite 451
- „num_poolagents - Agentenpoolgröße" auf Seite 451

agentpri - Agentenpriorität

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 [41 - 125]

Andere UNIX-Plattformen
-1 [41 - 128]

Windows
-1 [0 - 6]

Mit diesem Parameter wird die Priorität gesteuert, die sowohl allen Agenten als auch anderen Prozessen und Threads des Datenbankmanagerexemplars vom Scheduler des Betriebssystems zugewiesen wird. In einer Umgebung mit partitionierten Datenbanken gehören dazu auch koordinierende Agenten und Subagenten, parallele Systemsteuerprogramme und die FCM-Dämonen. Durch diese Priorität wird festgelegt, wie den DB2-Prozessen, -Agenten und -Threads im Vergleich zu den

anderen Prozessen und Threads, die auf dem System ausgeführt werden, CPU-Zeit zugewiesen wird. Wenn der Parameter auf den Wert -1 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 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 auf Ihrem System ausgeführt werden.

Mit diesem Parameter kann der Durchsatz des Datenbankmanager erhöht werden. Die Werte für diesen Parameters sind von dem Betriebssystem abhängig, auf dem der Datenbankmanager ausgeführt wird. Beispielsweise ergeben in einer auf UNIX basierenden 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 auf UNIX basierenden 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 erhalten. Sie sollten den Wert für diesen Parameter mit den anderen Aktivitäten, die Sie auf der Maschine erwarten, abstimmen.

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.

Anmerkung: Wenn Sie auf UNIX-Plattformen für diesen Parameter einen anderen als den Standardwert verwenden, können Sie Governor nicht verwenden, um Agentenprioritäten zu ändern.

Zugehörige Referenzen:

- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

avg_appls - Durchschnittliche Anzahl aktiver Anwendungen

Konfigurationstyp	Datenbank
Parametertyp	Online konfigurierbar
Weitergabeklasse	Anweisungsgrenzwert
Standardwert [Bereich]	1 [1 – maxappls]
Maßeinheit	Zähler

Mit Hilfe dieses Parameters versucht das SQL-Optimierungsprogramm zu ermitteln, wie viel Pufferpool zur Laufzeit für den ausgewählten Zugriffsplan verfügbar ist.

Empfehlung: Wenn DB2 in einer Mehrplatzsystemumgebung ausgeführt wird, kann es vorteilhaft sein, wenn das SQL-Optimierungsprogramm weiß, dass mehrere Abfragebenutzer das System verwenden (besonders bei komplexen Abfragen und einem großen Pufferpool), so dass das Optimierungsprogramm in seinen Annahmen zur Verfügbarkeit von Pufferpool weniger großzügig ist.

Bei der Einstellung dieses Parameters sollten Sie die Anzahl der Anwendungen mit komplexen Abfragen berechnen, die die Datenbank durchschnittlich verwenden. Aus dieser Berechnung sollten alle einfachen OLTP-Anwendungen (Online-Transaktionsprogramme) ausgeklammert werden. Wenn die Bestimmung dieser Anzahl schwierig ist, können Sie folgende Werte miteinander multiplizieren:

- Eine Durchschnittsanzahl aller Anwendungen, die für Ihre Datenbank ausgeführt werden. Der Datenbanksystemmonitor kann Informationen zur Anzahl der Anwendungen zu einem gegebenen Zeitpunkt liefern, so dass Sie anhand mehrerer Probewerte die Durchschnittsanzahl über einen gewissen Zeitraum hinweg berechnen können. Die Informationen des Datenbanksystemmonitors umfassen sowohl OLTP- als auch Nicht-OLTP-Anwendungen.
- Den von Ihnen geschätzten Prozentsatz an Anwendungen mit komplexen Abfragen.

Wie bei der Anpassung anderer Konfigurationsparameter, die das Optimierungsprogramm beeinflussen, sollten Sie auch den Wert dieses Parameters in kleinen Schritten ändern. Dadurch können Sie die Differenzen bei der Pfadauswahl minimieren.

Wenn Sie diesen Parameter geändert haben, sollten Sie Anwendungen eventuell erneut binden (mit dem Befehl REBIND PACKAGE).

Zugehörige Referenzen:

- „maxappls - Maximale Anzahl aktiver Anwendungen" auf Seite 447
- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

max_connections - Maximale Anzahl von Clientverbindungen

Konfigurationstyp	Datenbankmanager
Parametertyp	Konfigurierbar
Standardwert [Bereich]	-1 (*max_coordagents*) [-1; *max_coordagents* — 64 000]

Wenn der Konzentrator inaktiviert ist, gibt dieser Parameter die maximal zulässige Anzahl Clientverbindungen pro Partition an. Der Konzentrator ist inaktiviert, wenn *max_connections* gleich *max_coordagents* ist. Der Konzentrator ist aktiviert, wenn *max_connections* größer als *max_coordagents* ist.

Dieser Parameter steuert die maximale Anzahl der Anwendungen, die mit dem Exemplar verbunden sein können. In der Regel wird jede Anwendung einem Koordinatoragent zugeordnet. Ein Agent erleichtert 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 mit seinem eigenen privaten Speicher und benutzt 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 übersteigt. Es ist die Aufgabe des Konzentrators, die Anzahl der Serverressourcen pro Clientanwendung so weit zu verringern, dass ein DB2 Connect-Gateway mehr als 10 000 Clientverbindungen verarbeiten kann.

Der Wert -1 gibt an, dass der Grenzwert gleich dem Wert des Parameters *max_coordagents* ist.

In früheren Versionen von DB2 hatte dieser Parameter den Namen *max_logicagents*.

Zugehörige Referenzen:

- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

max_coordagents - Maximale Anzahl koordinierender Agenten

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 (*maxagents – num_initagents*)

[-1, 0–*maxagents*]

In Umgebungen mit partitionierten Datenbanken und Umgebungen, in denen *intra_parallel* auf "Yes" gesetzt ist, ist der Standardwert *maxagents* minus *num_initagents*; andernfalls ist der Standardwert *maxagents*. Dadurch wird sichergestellt, dass in Datenbankumgebungen ohne partitionierte Datenbanken *max_coordagents* immer gleich *maxagents* ist, außer wenn das System für partitionsinterne Parallelität konfiguriert ist.

Wenn Sie nicht mit einer Umgebung mit partitionierten Datenbanken arbeiten und den Parameter *intra_parallel* nicht aktiviert haben, muss *max_coordagents* gleich *maxagents* sein.

Wenn die Konzentratorfunktion nicht aktiviert ist, d. h., wenn *max_connections* gleich *max_coordagents* ist, legt dieser Parameter die maximale Anzahl koordinierender Agenten fest, die in einer partitionierten oder nicht partitionierten Datenbankumgebung gleichzeitig auf einem Server vorhanden sein können.

Für jede lokale oder ferne Anwendung, die eine Verbindung zu einer Datenbank oder einem Exemplar herstellt, wird ein koordinierender Agent aufgerufen. Anforderungen, für die eine Exemplarverbindung erforderlich ist, sind z. B. CREATE DATABASE, DROP DATABASE und Befehle des Datenbanksystemmonitors.

Wenn die Konzentratorfunktion 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.

Zugehörige Referenzen:

- „num_initagents - Anfangswert für die Anzahl Agenten im Pool" auf Seite 451
- „num_poolagents - Agentenpoolgröße" auf Seite 451
- „intra_parallel - Partitionsinterne Parallelität aktivieren" auf Seite 522
- „maxagents - Maximale Anzahl von Agenten" auf Seite 446
- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

maxagents - Maximale Anzahl von Agenten

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

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. 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 die Initialisierung und Verwaltung jedes weiteren Agenten ist zusätzlicher Speicher erforderlich, der beim Starten des Datenbankmanagers zugeordnet wird.

Zugehörige Referenzen:

- „max_coordagents - Maximale Anzahl koordinierender Agenten" auf Seite 445
- „num_poolagents - Agentenpoolgröße" auf Seite 451
- „maxcagents - Maximale Anzahl gleichzeitig aktiver Agenten" auf Seite 448
- „fenced_pool - Maximale Anzahl abgeschirmter Prozesse" auf Seite 453
- „maxappls - Maximale Anzahl aktiver Anwendungen" auf Seite 447
- „min_priv_mem - Mindestgröße des reservierten privaten Speichers" auf Seite 413
- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

maxappls - Maximale Anzahl aktiver Anwendungen

Konfigurationstyp	Datenbank
Parametertyp	Online konfigurierbar
Weitergabeklasse	Sofort
Standardwert [Bereich]	Automatic [Automatic; 1 – 60 000]
Maßeinheit	Zähler

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.

Wenn Sie den Parameter *maxappls* auf *automatic* setzen, ermöglichen Sie dadurch eine beliebige Anzahl verbundener Anwendungen. DB2 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 gleichzeitig im Prozess einer zweiphasigen COMMIT- oder einer ROLLBACK-Operation begriffen sein 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.

Wenn mehr Anwendungen Data Links Manager verwenden, sollte der Wert für *maxappls* erhöht werden. Berechnen Sie den erforderlichen Wert anhand folgender Formel:

```
<maxappls> = 5 * (Anzahl Knoten) + (Höchstanzahl aktiver Anwendungen,
                  die Data Links Manager verwenden)
```

Der unterstützte Maximalwert für Data Links Manager ist 2 000.

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 (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 *maxagents* bestimmt. Eine Anwendung kann nur dann eine Verbindung zur Datenbank herstellen, wenn eine freie Verbindung (*maxappls*) sowie ein freier Agent (*maxagents*) verfügbar ist. Darüber hinaus wird die maximale Anzahl von Anwendungen auch vom Konfigurationsparameter *max_coordagents* gesteuert, weil keine neuen Anwendungen (d. h. Koordinatoragenten) gestartet werden können, wenn der Wert von *max_coordagents* erreicht ist.

Zugehörige Tasks:

- „Manuelles Auflösen unbestätigter Transaktionen" in *Systemverwaltung: Konzept*

Zugehörige Referenzen:

- „max_coordagents - Maximale Anzahl koordinierender Agenten" auf Seite 445
- „maxagents - Maximale Anzahl von Agenten" auf Seite 446
- „locklist - Maximaler Speicher für Sperrenliste" auf Seite 399
- „maxlocks - Maximale Anzahl Sperren pro Anwendung" auf Seite 433
- „avg_appls - Durchschnittliche Anzahl aktiver Anwendungen" auf Seite 443
- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

maxcagents - Maximale Anzahl gleichzeitig aktiver Agenten

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

Mit diesem Parameter wird die maximale Anzahl von Datenbankmanageragenten festgelegt, die gleichzeitig eine Datenbankmanagertransaktion ausführen können. Dieser Parameter wird zur Steuerung der Systembelastung in Phasen hoher gleichzeitiger Anwendungsaktivität verwendet. 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.

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.

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, wo es durch den umfassenden gleichzeitigen Zugriff von Anwendungen zu Problemen kommt, können Sie durch Vergleichstests (Benchmark-Tests) die beste Einstellung für diesen Parameter ermitteln, um die Leistung der Datenbank zu optimieren.

Zugehörige Referenzen:

- „max_coordagents - Maximale Anzahl koordinierender Agenten" auf Seite 445
- „maxagents - Maximale Anzahl von Agenten" auf Seite 446
- „maxappls - Maximale Anzahl aktiver Anwendungen" auf Seite 447
- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

maxfilop - Maximale Anzahl offener Datenbankdateien pro Anwendung

Konfigurationstyp Datenbank

Parametertyp Online konfigurierbar

Weitergabeklasse Transaktionsgrenzwert

Standardwert [Bereich]

UNIX 64 [2 – 1950]

	Windows	64 [2 – 32 768]
Maßeinheit	Zähler	

Mit diesem Parameter wird die maximale Anzahl der Dateikennungen angegeben, die für jeden einzelnen Datenbankagenten geöffnet sein können. Wenn durch das Öffnen einer Datei dieser Wert überschritten wird, werden einige von diesem Agenten verwendete Dateien geschlossen. Wenn der Wert für den Parameter *maxfilop* zu klein ist, wächst der Systemaufwand für das Öffnen und Schließen von Dateien, um diesen Grenzwert nicht zu überschreiten, 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, so dass Dateikennungen für sie erforderlich sind. In der Regel werden von SMS-Tabellenbereichen vergleichsweise mehr Dateien verwendet, als von DMS-Dateitabellenbereichen Behälter verwendet werden. Daher wird für diesen Parameter ein höherer Wert benötigt, wenn SMS-Tabellenbereiche verwendet werden, als für DMS-Dateitabellenbereiche erforderlich wäre.

Mit Hilfe 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 Agent auf einen bestimmten Wert gesetzt wird. Die tatsächliche Anzahl ist je nach Anzahl der gleichzeitig aktiven Agenten unterschiedlich.

Zugehörige Referenzen:

- „maxappls - Maximale Anzahl aktiver Anwendungen" auf Seite 447
- „maxtotfilop - Maximale Anzahl offener Dateien" auf Seite 450
- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

maxtotfilop - Maximale Anzahl offener Dateien

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]	16 000 [100 – 32 768]
Maßeinheit	Zähler

Mit diesem Parameter wird die maximale Anzahl von Dateien definiert, die von allen Agenten und anderen Threads, die in einem Datenbankmanagerexemplar ausgeführt werden, geöffnet werden können. Wenn durch das Öffnen einer Datei der Wert dieses Parameters überschritten wird, wird ein Fehler an die Anwendung zurückgegeben.

Anmerkung: Dieser Parameter gilt nicht für auf UNIX basierende Plattformen.

Empfehlung: Bei der Einstellung dieses Parameters sollte die Anzahl der Dateikennungen beachtet werden, die für jede Datenbank im Datenbankmanagerexemplar verwendet werden könnten. Ermitteln Sie auf folgende Weise einen oberen Grenzwert für diesen Parameter:

1. Berechnen Sie anhand der folgenden Formel die maximale Anzahl von Dateikennungen, die für jede Datenbank im Exemplar geöffnet werden könnten:

   ```
   maxappls * maxfilop
   ```

2. Berechnen Sie die Summe der obigen Ergebnisse, und stellen Sie sicher, dass sie den Maximalwert des Parameters nicht überschreitet.

Wenn eine neue Datenbank erstellt wird, sollten Sie den Wert für diesen Parameter neu bestimmen.

Zugehörige Referenzen:

- „maxfilop - Maximale Anzahl offener Datenbankdateien pro Anwendung" auf Seite 449
- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

num_initagents - Anfangswert für die Anzahl Agenten im Pool

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 — *num_poolagents*]

Dieser Parameter gibt an, wie viele inaktive Agenten beim Ausführen von DB2START im Agentenpool erstellt werden.

Zugehörige Referenzen:

- „max_coordagents - Maximale Anzahl koordinierender Agenten" auf Seite 445
- „num_poolagents - Agentenpoolgröße" auf Seite 451
- „maxagents - Maximale Anzahl von Agenten" auf Seite 446
- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

num_poolagents - Agentenpoolgröße

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 (*maxagents* / 2) [-1; 0 — *maxagents*]

Wenn die Konzentratorfunktion nicht aktiviert ist, d. h., wenn *max_connections* gleich *max_coordagents* ist, legt dieser Parameter die maximale Größe des Pools inaktiver Agenten fest. Inaktive Agenten können als parallele Subagenten oder als Koordinatoragenten verwendet werden. Wurden mehr Agenten erstellt, als der Wert dieses Parameters angibt, werden diese nach Ausführung ihrer aktuellen Anforderung nicht in den Pool zurückgestellt sondern beendet.

Wenn die Konzentratorfunktion aktiviert ist, d. h., wenn *max_connections* größer als *max_coordagents* ist, werden Agenten unabhängig vom Wert dieses Parameters immer an den Pool zurückgegeben. Basierend auf der Systembelastung und der Zeit, die die Agenten inaktiv im Pool bleiben, können sich Agenten bei Bedarf selbst beenden, um die Größe des Pools inaktiver Agenten auf den konfigurierten Parameterwert zu reduzieren.

Außer bei aktivierter Konzentratorfunktion werden, wenn dieser Parameter den Wert 0 hat, Agenten nach Bedarf erstellt und beendet, wenn sie die Ausführung ihrer aktuellen Anforderung abgeschlossen haben.

Empfehlung: Wenn Sie in einer Entscheidungshilfeumgebung arbeiten, in der nur wenige Anwendungen gleichzeitig mit der Datenbank verbunden sind, setzen Sie den Parameter *num_poolagents* auf einen kleinen Wert, um zu verhindern, dass ein Agentenpool mit inaktiven Agenten gefüllt wird.

Wenn Sie eine Transaktionsverarbeitungsumgebung betreiben, in der viele Anwendungen gleichzeitig mit der Datenbank verbunden sind, erhöhen Sie den Wert für *num_poolagents*, um den Aufwand für die häufige Erstellung und Beendigung von Agenten zu vermeiden.

Zugehörige Referenzen:

- „num_initagents - Anfangswert für die Anzahl Agenten im Pool" auf Seite 451
- „max_coordagents - Maximale Anzahl koordinierender Agenten" auf Seite 445
- „max_querydegree - Maximaler Grad der Parallelität bei Abfragen" auf Seite 522
- „maxagents - Maximale Anzahl von Agenten" auf Seite 446
- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

Gespeicherte Prozeduren und benutzerdefinierte Funktionen

Die folgenden Parameter können sich auf die Leistung abgeschirmter gespeicherter Prozeduren und benutzerdefinierter Funktionen auswirken:

- „fenced_pool - Maximale Anzahl abgeschirmter Prozesse" auf Seite 453
- „keepfenced - Abgeschirmten Prozess beibehalten" auf Seite 454

- „num_initfenced - Anfangswert für die Anzahl abgeschirmter Prozesse” auf Seite 455

fenced_pool - Maximale Anzahl abgeschirmter Prozesse

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*)

Maßeinheit Zähler

Für Thread-db2fmp-Prozesse (Prozesse, die threadsichere gespeicherte Prozeduren und benutzerdefinierte Funktionen bedienen) stellt dieser Parameter die Anzahl Threads dar, die in jedem db2fmp-Prozess zwischengespeichert werden. Für db2fmp-Prozesse, die keine Threadprozesse sind, stellt dieser Parameter die Anzahl zwischengespeicherter 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 dem Exemplar 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 der Wert dieses Parameters auf –1 gesetzt wird, ist die maximale Anzahl von Prozessen im abgeschirmten Modus gleich dem für den Parameter *max_coordagents* festgelegten Wert.

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 Datenbankmanager kommt, kann anhand der folgenden Informationen eine erste Optimierung für diesen Parameters vorgenommen werden:

```
fenced_pool = Anzahl Anwendungen, die gleichzeitig Aufrufe gespeicherter Prozeduren und
   UDF-Aufrufe absetzen dürfen
```

Wenn der Parameter *keepfenced* auf den Wert *yes* gesetzt ist, bleibt jeder im Cachepool erstellte db2fmp-Prozess bestehen und verbraucht auch dann noch Systemressourcen, wenn der Aufruf der Routine im abgeschirmten Modus verarbeitet und an den Agenten zurückgegeben wurde.

Wenn *keepfenced* auf *no* gesetzt ist, werden db2fmp-Prozesse, die keine Threadprozesse sind, nach Beendigung der Ausführung beendet, und es gibt keinen Cachepool. Multithread-db2fmp-Prozesse sind weiter vorhanden, es werden in diesen Prozessen jedoch keine Threads in den Pool gestellt. Dies bedeutet, das Sie auf Ihrem System einen C-Thread-db2fmp-Prozess und einen Java-Thread-db2fmp-Prozess haben können, auch wenn *keepfenced* auf *no* gesetzt ist.

In früheren Versionen von DB2 hatte dieser Parameter den Namen *maxdari*.

Zugehörige Referenzen:

- „max_coordagents - Maximale Anzahl koordinierender Agenten" auf Seite 445
- „maxagents - Maximale Anzahl von Agenten" auf Seite 446
- „keepfenced - Abgeschirmten Prozess beibehalten" auf Seite 454
- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „num_initfenced - Anfangswert für die Anzahl abgeschirmter Prozesse" auf Seite 455

keepfenced - Abgeschirmten Prozess beibehalten

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 [Yes; No]

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.

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 der zusätzliche Systemaufwand 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, entsteht für jede Transaktion der Systemaufwand zum Erstellen eines neuen Prozesses im abgeschirmten Modus, 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 der zusätzliche Systemaufwand vermieden wird.

In früheren Versionen von DB2 hatte dieser Parameter den Namen *keepdari*.

Zugehörige Referenzen:

- „fenced_pool - Maximale Anzahl abgeschirmter Prozesse" auf Seite 453
- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

num_initfenced - Anfangswert für die Anzahl abgeschirmter Prozesse

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 — *max_connections* + (*maxagents* - *max_coordagents*)]

Dieser Parameter gibt die Anfangszahl der inaktiven db2fmp-Prozesse an, die keine Threadprozesse sind und die zum Startzeitpunkt der Datenbank (DB2START) im db2fmp-Pool erstellt werden. 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 der Datenbank (DB2START) eine Reihe von db2fmp-Prozessen zu starten.

In früheren Versionen von DB2 hatte dieser Parameter den Namen *num_initdaris*.

Zugehörige Referenzen:

- „keepfenced - Abgeschirmten Prozess beibehalten" auf Seite 454
- „fenced_pool - Maximale Anzahl abgeschirmter Prozesse" auf Seite 453
- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

Protokollieren und Wiederherstellung

Die Wiederherstellung der Umgebung kann eine wichtige Maßnahme sein, um den Verlust kritischer Daten zu verhindern. Es steht eine Reihe von Parametern zur Verfügung, die Ihnen bei der Verwaltung der Umgebung helfen, um sicherstellen zu können, dass eine geeignete Wiederherstellung der Daten bzw. Transaktionen durchgeführt werden kann. Diese Parameter werden in folgende Kategorien unterteilt:

- „Datenbankprotokolldateien"
- „Datenbankprotokollierung" auf Seite 467
- „Wiederherstellung" auf Seite 477
- „Wiederherstellung verteilter Arbeitseinheiten" auf Seite 488

Datenbankprotokolldateien

Die folgenden Parameter enthalten Informationen zu Anzahl, Größe und Status der Dateien, die zur Datenbankprotokollierung verwendet werden:

- „logfilsiz - Protokolldateigröße"
- „loghead - Erste aktive Protokolldatei" auf Seite 458
- „logpath - Pfad zu Protokolldateien" auf Seite 458
- „logprimary - Anzahl primärer Protokolldateien" auf Seite 458
- „logsecond - Anzahl sekundärer Protokolldateien" auf Seite 460
- „max_log - Maximuale Protokollgröße pro Transaktion" auf Seite 462
- „mirrorlogpath - Pfad für Protokollspiegelung" auf Seite 462
- „newlogpath - Datenbankprotokollpfad ändern" auf Seite 463
- „num_log_span - Anzahl verwendeter Protokolldateien" auf Seite 465
- „overflowlogpath - Überlaufprotokollpfad" auf Seite 465

logfilsiz - Protokolldateigröße

Konfigurationstyp Datenbank

Parametertyp Konfigurierbar

Standardwert [Bereich]

UNIX 1000 [4 — 262 144]

	Windows	1000 [4 — 262 144]
	Maßeinheit	Seiten (4 KB)

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.

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 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.
- Protokollierung mit Protokollspeicherung

  Wenn eine primäre Protokolldatei voll ist, wird das Protokoll archiviert und eine neue primäre Protokolldatei zugeordnet.

Empfehlung: Die Größe der Protokolldateien muss mit der Anzahl der primären 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 für den aktiven Protokollspeicherbereich einen oberen Grenzwert von 256 GB.

  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 einen erhöhten Systemaufwand mit sich bringen.
- 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.

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.

Zugehörige Referenzen:

- „logprimary - Anzahl primärer Protokolldateien" auf Seite 458
- „logsecond - Anzahl sekundärer Protokolldateien" auf Seite 460
- „softmax - Vor bedingtem Prüfpunkt zu schreibende Protokollsätze" auf Seite 474
- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

loghead - Erste aktive Protokolldatei

Konfigurationstyp Datenbank

Parametertyp Informativ

Dieser Parameter gibt den Namen der zurzeit aktiven Protokolldatei an.

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*

logpath - Pfad zu Protokolldateien

Konfigurationstyp Datenbank

Parametertyp Informativ

Dieser Parameter gibt den aktuellen Pfad an, der für die Protokolldateien verwendet wird. Dieser Parameter kann nicht direkt geändert werden, da der Wert vom Datenbankmanager festgelegt wird, wenn eine Änderung am Parameter *newlogpath* wirksam wird.

Bei der Erstellung einer Datenbank wird die zugehörige Datei des Wiederherstellungsprotokolls in einem Unterverzeichnis des Verzeichnisses erstellt, in dem sich die Datenbank befindet. Standardmäßig wird dieses Unterverzeichnis mit dem Namen SQLOGDIR in dem Verzeichnis angelegt, das für die Datenbank erstellt wurde.

Zugehörige Referenzen:

- „newlogpath - Datenbankprotokollpfad ändern" auf Seite 463
- „GET DATABASE CONFIGURATION Command" in *Command Reference*

logprimary - Anzahl primärer Protokolldateien

Konfigurationstyp Datenbank

Parametertyp Konfigurierbar

Standardwert [Bereich] 3 [2 – 256]

Maßeinheit Zähler

Zuordnung

- Wenn die Datenbank erstellt wird.
- Wenn ein Protokoll an eine andere Speicherposition versetzt wird (dies geschieht, wenn der Parameter *logpath* aktualisiert wird).
- Wenn nach der Erhöhung des Werts für den Parameter *logprimary* die nächste Verbindung zur

Datenbank hergestellt wird, nachdem alle Benutzer die Verbindung getrennt hatten.

- Wenn nach der Archivierung einer Protokolldatei eine neue Protokolldatei zugeordnet wird. (Der Parameter *logretain* oder *userexit* muss aktiviert sein.)
- Wenn der Parameter *logfilsiz* geändert wurde, wird für die aktiven Protokolldateien die neue Größe aktiv, wenn die nächste Verbindung zur Datenbank hergestellt wird, nachdem alle Benutzer die Verbindung zur Datenbank getrennt hatten.

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.

Die primären Protokolldateien reservieren eine feste Menge Speicher, der den Wiederherstellungsprotokollen zugeordnet wird. Mit diesem Parameter kann die Anzahl der im Voraus zugeordneten primären Protokolldateien festgelegt werden.

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 Arbeitseinheiten mit Protokollsätzen in diesem Protokoll festgeschrieben oder 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 des Festschreibens) 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 Wiederherstellung aktiviert ist, sollte dieser Parameter auf einen größeren Wert gesetzt werden, um den erhöhten Systemaufwand aufgrund der beinahe sofort erforderlichen 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 (Max. verwendeter sekundärer Protokollbereich)*
- *tot_log_used_top (Max. verwendeter Gesamtbereich für Protokolle)*
- *sec_logs_allocated (Momentan zugeordnete sekundäre Protokolle)*

Zugehörige Referenzen:

- „logfilsiz - Protokolldateigröße" auf Seite 456
- „logsecond - Anzahl sekundärer Protokolldateien" auf Seite 460
- „logretain - Beibehalten von Protokollen aktivieren" auf Seite 471
- „userexit - Benutzerexit aktivieren" auf Seite 475
- „sec_log_used_top - Maximum Secondary Log Space Used monitor element" in *System Monitor Guide and Reference*
- „tot_log_used_top - Maximum Total Log Space Used monitor element" in *System Monitor Guide and Reference*
- „sec_logs_allocated - Secondary Logs Allocated Currently monitor element" in *System Monitor Guide and Reference*
- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

logsecond - Anzahl sekundärer Protokolldateien

Konfigurationstyp	Datenbank
Parametertyp	Online konfigurierbar
Weitergabeklasse	Sofort
Standardwert [Bereich]	2 [-1; 0 – 254]
Maßeinheit	Zähler
Zuordnung	Nach Bedarf, wenn *logprimary* nicht ausreichend ist (siehe unten).
Freigabe	Wenn der Datenbankmanager feststellt, dass sie nicht mehr erforderlich sind.

Dieser Parameter gibt die Anzahl der sekundären Protokolldateien an, die (nach Bedarf) erstellt und für Wiederherstellungsprotokolle verwendet werden. 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 sekun-

däre Protokolldateien erforderlich sind, als von diesem Parameter zugelassen werden, wird ein Fehlercode an die Anwendung zurückgegeben, und die Datenbank wird beendet.

Wenn Sie *logsecond* auf -1 setzen, wird die Datenbank mit unbegrenztem aktiven Speicherbereich konfiguriert. Für die Größe oder Anzahl der unvollständigen Transaktionen, die in der Datenbank ausgeführt werden, gibt es keine Begrenzung. Wenn Sie *logsecond* auf -1 setzen, verwenden Sie weiter die Konfigurationsparameter *logprimary* und *logfilsiz*, um die Anzahl der Protokolldateien anzugeben, die DB2 im Pfad für aktive Protokolldateien behalten soll. Wenn DB2 Protokolldaten aus einer Protokolldatei lesen muss, die sich nicht im Pfad für aktive Protokolldateien befindet, ruft DB2 das Benutzerexitprogramm auf, um die Protokolldatei aus dem Archiv in den Pfad für aktive Protokolldateien abzurufen. (DB2 ruft die Dateien in den Überlaufprotokollpfad ab, falls Sie einen solchen Pfad konfiguriert haben.) Nach dem Abrufen der Protokolldatei wird diese Datei von DB2 im Pfad für aktive Protokolldateien zwischengespeichert, so dass das Lesen weiterer Protokolldaten aus dieser Datei schneller ablaufen kann. DB2 verwaltet das Abrufen, Zwischenspeichern und Entfernen dieser Protokolldateien nach Bedarf.

Wenn Ihr Protokollpfad eine unformatierte Einheit ist, müssen Sie den Konfigurationsparameter *overflowlogpath* konfigurieren, um *logsecond* auf -1 zu setzen.

Wenn Sie *logsecond* auf -1 setzen, besteht für die Größe der Arbeitseinheit oder die Anzahl gleichzeitig ablaufender Arbeitseinheiten keine Begrenzung. Allerdings können ROLLBACK-Operationen (sowohl auf Sicherungspunktebene als auch auf Arbeitseinheitenebene) viel Zeit in Anspruch nehmen, da Protokolldateien aus dem Archiv abgerufen werden müssen. Aus demselben Grund kann die Wiederherstellung nach einem Systemabsturz ebenfalls sehr zeitintensiv sein. DB2 schreibt eine Nachricht in das Benachrichtigungsprotokoll für die Verwaltung, um Sie darauf hinzuweisen, dass die aktuelle Menge aktiver Arbeitseinheiten die primären Protokolldateien überschreitet. Dies deutet ebenfalls an, dass ROLLBACK-Operationen oder die Wiederherstellung nach einem Systemabsturz viel Zeit in Anspruch nehmen können.

Damit *logsecond* auf -1 gesetzt werden kann, muss der Konfigurationsparameter *userexit* auf *yes* gesetzt sein.

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.

Zugehörige Referenzen:

- „logfilsiz - Protokolldateigröße" auf Seite 456
- „logprimary - Anzahl primärer Protokolldateien" auf Seite 458
- „logretain - Beibehalten von Protokollen aktivieren" auf Seite 471
- „userexit - Benutzerexit aktivieren" auf Seite 475
- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*

- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*
- „overflowlogpath - Überlaufprotokollpfad" auf Seite 465

max_log - Maximuale Protokollgröße pro Transaktion

Konfigurationstyp	Datenbank
Parametertyp	Online konfigurierbar
Weitergabeklasse	Sofort
Standardwert [Bereich]	0 [0 — 100]
Maßeinheit	Prozent

Wenn der Wert nicht 0 ist, zeigt dieser Parameter den Prozentsatz des aktiven Protokollspeicherbereichs an, der von einer Transaktion verbraucht werden kann.

Wird der Wert auf 0 gesetzt, besteht keine Begrenzung des Prozentsatzes des gesamten aktiven Protokollbereichs, den eine Transaktion verbrauchen kann. Dies war die Funktionsweise von Transaktionen vor Version 8.

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*
- „num_log_span - Anzahl verwendeter Protokolldateien" auf Seite 465

mirrorlogpath - Pfad für Protokollspiegelung

Konfigurationstyp	Datenbank
Parametertyp	Konfigurierbar
Standardwert [Bereich]	Null [gültiger Pfad oder gültige Einheit]

Mit diesem Parameter können Sie eine Zeichenfolge von bis zu 242 Byte für den Pfad für die Protokollspiegelung angeben. Diese Zeichenfolge muss auf einen Pfadnamen verweisen, der ein vollständig qualifizierter Pfadname und kein relativer Pfadname ist.

Anmerkung: In einer Umgebung mit partitionierten Datenbanken wird die Knotennummer automatisch an den Pfad angefügt. Dadurch wird die Eindeutigkeit des Pfads in Konfigurationen mit mehreren logischen Knoten sichergestellt.

Wenn *mirrorlogpath* konfiguriert ist, erstellt DB2 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, so dass die Datenbank im Fall eines Plattenfehlers oder eines Benutzerfehlers, durch den aktive Protokolldateien in einem der Pfade zerstört werden, weiter funktioniert.

Wenn der Pfad für die Protokollspiegelung geändert wird, können sich im alten Pfad für die Protokollspiegelung immer noch Protokolldateien befinden. Diese Protokolldateien wurden eventuell nicht archiviert, so dass Sie sie möglicherweise manuell archivieren müssen. Wenn Sie zudem für diese Datenbank Replikation ausführen, benötigt Replikation eventuell weiterhin die vor der Protokollpfadänderung vorhandenen Protokolldateien. Wenn der Datenbankkonfigurationspara-

meter für Benutzer-Exit (*userexit*) für die Datenbank auf ″Yes″ gesetzt ist und wenn alle Protokolldateien entweder automatisch durch DB2 oder von Ihnen selbst manuell archiviert wurden, dann kann DB2 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 *logpath* oder *newlogpath* eine unformatierte Einheit als Position zum Speichern der Protokolldateien angibt, ist die Protokollspiegelung, die durch *mirrorlogpath* angegeben wird, nicht zulässig. Wenn *logpath* oder *newlogpath* den Dateipfad als Position zum Speichern von Protokolldateien angegeben wird, ist die Protokollspiegelung zulässig, und *mirrorlogpath* muss ebenfalls einen Dateipfad angeben.

Empfehlung: Wie die Protokolldateien sollten sich auch die Spiegelprotokolldateien auf einem physischen Laufwerk mit nur geringem E/A befinden.

Es wird dringend empfohlen, dass dieser Pfad auf einer anderen Einheit eingerichtet wird, als der primäre Protokollpfad.

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. 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.

- *log_reads* (Anzahl gelesener Protokollseiten)
- *log_writes* (Anzahl geschriebener Protokollseiten)

Zugehörige Referenzen:

- „logpath - Pfad zu Protokolldateien″ auf Seite 458
- „newlogpath - Datenbankprotokollpfad ändern″ auf Seite 463
- „log_reads - Number of Log Pages Read monitor element″ in *System Monitor Guide and Reference*
- „log_writes - Number of Log Pages Written monitor element″ in *System Monitor Guide and Reference*
- „GET DATABASE CONFIGURATION Command″ in *Command Reference*
- „RESET DATABASE CONFIGURATION Command″ in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command″ in *Command Reference*
- „overflowlogpath - Überlaufprotokollpfad″ auf Seite 465

newlogpath - Datenbankprotokollpfad ändern

Konfigurationstyp	Datenbank
Parametertyp	Konfigurierbar
Standardwert [Bereich]	Null [gültiger Pfad oder gültige Einheit]

Mit diesem Parameter können Sie eine Zeichenfolge mit bis zu 242 Byte angeben, um die Speicherposition der Protokolldateien zu ändern. Die Zeichenfolge kann auf einen Pfadnamen oder auf eine unformatierte Einheit verweisen. Verweist die Zeichenfolge auf einen Pfadnamen, muss es sich um einen vollständig qualifizierten Pfad handeln und nicht um einen relativen Pfad.

Anmerkung: In einer Umgebung mit partitionierten Datenbanken wird die Knotennummer automatisch an den Pfad angefügt. Dadurch wird die Eindeutigkeit des Pfads in Konfigurationen mit mehreren logischen Knoten sichergestellt.

Wenn Sie Replikation verwenden wollen und Ihr Protokollpfad eine unformatierte Einheit ist, muss der Konfigurationsparameter *overflowlogpath* konfiguriert werden.

Geben Sie zum Angeben einer Einheit eine Zeichenfolge an, die vom Betriebssystem als eine Einheit erkannt wird. Zum Beispiel:

- Unter Windows NT, `\\.\d:` oder `\\.\PhysicalDisk5`

  Anmerkung: Damit Protokolle in eine Einheit geschrieben werden können, muss Windows NT Version 4.0 mit Service Pack 3 oder höher installiert sein.
- Auf UNIX-Plattformen `/dev/rdblog8`

Anmerkung: Eine Einheit können Sie nur auf AIX-, Windows 2000-, Windows NT-, Solaris-Betriebsumgebung-, HP-UX- und Linux-Plattformen angeben.

Diese Einstellung wird nur dann zum Wert des Parameters *logpath* , wenn die beiden folgenden Bedingungen zutreffen:

- Die Datenbank ist in einem konsistenten Zustand, wie durch den Parameter *database_consistent* angegeben.
- Alle Benutzer 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 Replikation ausführen, benötigt Replikation eventuell weiterhin die vor der Protokollpfadänderung vorhandenen Protokolldateien. Wenn der Datenbankkonfigurationsparameter für Benutzerexit (*userexit*) für die Datenbank auf "Yes" gesetzt ist und wenn alle Protokolldateien entweder automatisch durch DB2 oder von Ihnen selbst manuell archiviert wurden, dann kann DB2 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 unformatierte Einheit als Position zum Speichern der Protokolldateien angibt, ist die Protokollspiegelung, die durch *mirrorlogpath* angegeben wird, nicht zulässig. Wenn *logpath* oder *newlogpath* den Dateipfad als Position zum Speichern von Protokolldateien angegeben wird, 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 **nicht** häufig E/A-Operationen auftreten. Sie sollten beispielsweise die Protokolldateien nicht auf derselben Platte wie das Betriebssystem oder umfangreiche Datenbanken speichern. Dadurch werden die Protokolliervorgänge effizient und verursachen nur ein Minimum an Systemaufwand, wie z. B. 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 die auf Datenbankprotokollierung bezogene Menge der 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.

Zugehörige Referenzen:

- „logpath - Pfad zu Protokolldateien" auf Seite 458
- „database_consistent - Datenbank ist konsistent" auf Seite 500
- „log_reads - Number of Log Pages Read monitor element" in *System Monitor Guide and Reference*
- „log_writes - Number of Log Pages Written monitor element" in *System Monitor Guide and Reference*
- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

num_log_span - Anzahl verwendeter Protokolldateien

Konfigurationstyp	Datenbank
Parametertyp	Online 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. Dies war die Funktionsweise von Transaktionen vor Version 8.

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*
- „max_log - Maximuale Protokollgröße pro Transaktion" auf Seite 462

overflowlogpath - Überlaufprotokollpfad

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 diesem Parameter können Sie eine Speicherposition angeben, an der DB2 nach Protokolldateien suchen soll, 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-Operation.
- 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, ruft DB2 die Protokolldateien in den Pfad für aktive Protokolldateien ab. Mit dem Parameter *overflowlogpath* können Sie DB2 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 (vor DB2 Version 8 hatte db2ReadLog den Namen sqlurlog) benutzen müssen, können Sie mit dem Parameter *overflowlogpath* eine Position angeben, an der DB2 nach Protokolldateien sucht, die für diese API benötigt werden. Wenn die Protokolldatei nicht gefunden werden kann (weder im Pfad für aktive Protokolldateien noch im Überlaufprotokollpfad) und in der Datenbankkonfiguration *userexit* aktiviert ist, ruft DB2 die Protokolldatei ab. Über den Parameter *overflowlogpath* können Sie auch ein Verzeichnis angeben, in dem DB2 die abgerufenen Protokolldateien speichert. 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 als Pfad für aktive Protokolldateien eine unformatierte Einheit konfiguriert haben, muss *overflowlogpath* konfiguriert werden, wenn Sie *logsecond* auf -1 setzen oder die API db2ReadLog verwenden wollen.

Wenn Sie *overflowlogpath* festlegen wollen, geben Sie eine Zeichenfolge von maximal 242 Byte an. Diese Zeichenfolge muss auf einen Pfadnamen verweisen, der ein vollständig qualifizierter Pfadname und kein relativer Pfadname ist. Der Pfadname muss ein Verzeichnis sein, keine unformatierte Einheit.

Anmerkung: In einer Umgebung mit partitionierten Datenbanken wird die Knotennummer automatisch an den Pfad angefügt. Dadurch wird die Eindeutigkeit des Pfads in Konfigurationen mit mehreren logischen Knoten sichergestellt.

Zugehörige Referenzen:

- „logsecond - Anzahl sekundärer Protokolldateien" auf Seite 460
- „db2ReadLog - Asynchronous Read Log" in *Administrative API Reference*
- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „ROLLFORWARD DATABASE Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

Datenbankprotokollierung

Die folgenden Parameter können den Typ und die Leistung der Datenbankprotokollierung beeinflussen:

- „archretrydelay - Wiederholungsintervall für Archivierung bei Fehler"
- „blk_log_dsk_ful - Bei voller Protokollplatte blockieren"
- „failarchpath - Protokollarchivpfad für Funktionsübernahme" auf Seite 468
- „logarchmeth1 - Primäre Protokollarchivierungsmethode" auf Seite 469
- „logarchmeth2 - Sekundäre Protokollarchivierungsmethode" auf Seite 469
- „logarchopt1 - Optionen für primäres Protokollarchiv" auf Seite 470
- „logarchopt2 - Optionen für sekundäres Protokollarchiv" auf Seite 470
- „logindexbuild - Erstellte Indexseiten protokollieren" auf Seite 470
- „logretain - Beibehalten von Protokollen aktivieren" auf Seite 471
- „mincommit - Anzahl der Gruppenfestschreibungen" auf Seite 471
- „numarchretry - Anzahl der Wiederholungen bei Fehler" auf Seite 473
- „softmax - Vor bedingtem Prüfpunkt zu schreibende Protokollsätze" auf Seite 474
- „userexit - Benutzerexit aktivieren" auf Seite 475
- „vendoropt - Lieferantenoptionen" auf Seite 476

archretrydelay - Wiederholungsintervall für Archivierung bei Fehler

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]

Mit diesem Parameter wird die Anzahl von Sekunden angegeben, die nach einem fehlgeschlagenen Archivierungsversuch abzuwarten ist, bevor die Archivierung der Protokolldatei erneut versucht wird. Nachfolgende Wiederholungen finden nur statt, wenn der Datenbankkonfigurationsparameter *numarchretry* mindestens auf den Wert 1 gesetzt ist.

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

blk_log_dsk_ful - Bei voller Protokollplatte blockieren

Konfigurationstyp Datenbank
Parametertyp Online konfigurierbar
Weitergabeklasse Sofort
Standardwert [Bereich] No [Yes; No]

Dieser Konfigurationsparameter kann so festgelegt werden, dass keine Fehler aufgrund erschöpfter Festplattenkapazität generiert werden, wenn DB2 keine neue Protokolldatei im aktiven Protokollpfad erstellen kann. Stattdessen wird DB2 alle fünf Minuten versuchen, die Protokolldatei zu erstellen, bis das Erstellen erfolgreich ist. Nach jedem Versuch schreibt DB2 eine Nachricht in das Protokoll mit 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. Solange die Protokolldatei noch nicht erstellt werden konnte, kann keine der Benutzeranwendungen, die versucht, Tabellendaten zu aktualisieren, Transaktionen 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, blockieren Anwendungen, sobald DB2 einen Fehler aufgrund erschöpfter Festplattenkapazität feststellt. Dadurch können Sie den Fehler beheben und die Transaktion anschließend beenden. Bei erschöpfter Festplattenkapazität können Sie beispielsweise alte Protokolldateien auf ein anderes Dateisystem versetzen oder das Dateisystem vergrößern, so dass blockierte Anwendungen beendet werden können.

Wenn *blk_log_dsk_ful* auf *no* gesetzt ist und eine Transaktion einen Fehler aufgrund erschöpfter Festplattenkapazität feststellt, schlägt die Transaktion fehl und wird rückgängig gemacht. In einigen Situationen kann die Datenbank abstürzen, wenn eine Transaktion einen Fehler aufgrund erschöpfter Festplattenkapazität verursacht.

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

failarchpath - Protokollarchivpfad für Funktionsübernahme

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 [ ]

Mit diesem Parameter wird ein Pfad angegeben, in dem DB2 versucht, Protokolldateien zu archivieren, wenn die Protokolldateien weder am primären noch am sekundären (falls definiert) Archivierungsziel archiviert werden können, weil diese Ziele von einem Datenträgerproblem betroffen sind. Der angegebene Pfad muss auf eine Platte verweisen.

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

logarchmeth1 - Primäre Protokollarchivierungsmethode

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] Off [ ]

Mit diesem Parameter wird der Datenträgertyp des primären Ziels für archivierte Protokolle angegeben.

Zugehörige Konzepte:

- Anhang E, „Knotenübergreifende Fehlerbehebung mit dem Befehl 'db2adutl' und den Datenbankkonfigurationsparametern 'logarchopt1' und 'vendoropt'", auf Seite 679

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

logarchmeth2 - Sekundäre Protokollarchivierungsmethode

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] Off [ ]

Mit diesem Parameter wird der Datenträgertyp des sekundären Ziels für archivierte Protokolle angegeben. Wenn dieser Pfad angegeben wird, werden Protokolldateien sowohl an diesem Ziel als auch an dem Ziel archiviert, das durch den Datenbankkonfigurationsparameter *logarchmeth1* angegeben wird.

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*

- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

logarchopt1 - Optionen für primäres Protokollarchiv

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 [ ]

Mit diesem Parameter wird das Feld der Optionen für das primäre Ziel für archivierte Protokolle (falls erforderlich) angegeben.

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

logarchopt2 - Optionen für sekundäres Protokollarchiv

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 [ ]

Mit diesem Parameter wird das Feld der Optionen für das sekundäre Ziel für archivierte Protokolle (falls erforderlich) angegeben.

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

logindexbuild - Erstellte Indexseiten protokollieren

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] Off [On; Off]

Mit diesem Parameter wird angegeben, ob Operationen zur Erstellung, erneuten Erstellung oder Reorganisation von Indizes protokolliert werden, so dass Indizes bei DB2-Operationen zur aktualisierenden Wiederherstellung oder bei Prozeduren zum Nachvollziehen von HADR-Protokollen (High Availability Disaster Recovery) rekonstruiert werden können.

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

logretain - Beibehalten von Protokollen aktivieren

Konfigurationstyp Datenbank

Parametertyp Konfigurierbar

Standardwert [Bereich] No [Recovery; No]

Es gibt folgende Werte:

- "No" gibt an, dass die Protokolle nicht beibehalten werden.
- "Recovery" gibt an, dass die Protokolle beibehalten werden und zur aktualisierenden Wiederherstellung verwendet werden können.

Ist *logretain* auf "Recovery" oder *userexit* auf "Yes" gesetzt, werden die aktiven Protokolldateien gespeichert und als Online-Archivprotokolldateien in einer aktualisierenden Wiederherstellung eingesetzt. Dies wird Protokollierung mit Protokollspeicherung genannt.

Nach dem Setzen von *logretain* auf "Recovery" und/oder von *userexit* auf "Yes" müssen Sie eine Gesamtsicherung der Datenbank vornehmen. Dieser Status wird durch den Markierungsparameter *backup_pending* angezeigt.

Ist *logretain* auf "No" und *userexit* auf "No" gesetzt, ist die aktualisierende Wiederherstellung für die Datenbank nicht verfügbar, da die Protokolle nicht beibehalten werden. In diesem Fall löscht der Datenbankmanager alle Protokolldateien im Verzeichnis *logpath* (einschließlich der Online-Archivprotokolldateien), ordnet er neue aktive Protokolldateien zu und aktiviert er erneut Umlaufprotokollierung.

Zugehörige Referenzen:

- „log_retain_status - Statusanzeiger für Beibehalten der Protokolle" auf Seite 500
- „userexit - Benutzerexit aktivieren" auf Seite 475
- „backup_pending - Sicherung anstehend" auf Seite 499
- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

mincommit - Anzahl der Gruppenfestschreibungen

Konfigurationstyp Datenbank

Parametertyp	Online konfigurierbar
Weitergabeklasse	Sofort
Standardwert [Bereich]	1 [1 – 25]
Maßeinheit	Zähler

Mit diesem Parameter können Sie das Schreiben von Protokollsätzen auf die Festplatte verzögern, bis eine Mindestanzahl von Festschreibungen (COMMIT-Operationen) ausgeführt wurde. Diese Verzögerung kann dazu beitragen, den Aufwand des Datenbankmanagers für das Schreiben von Protokollsätzen zu reduzieren. Dies führt zu einer Leistungssteigerung, wenn mehrere Anwendungen für eine Datenbank ausgeführt und von den Anwendungen viele Festschreibungen innerhalb kurzer Zeit angefordert werden.

Diese Gruppierung von Festschreibungen 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 Festschreibungen ausgeführt wird, können Festschreibungsanforderungen von Anwendungen so lange zurückgehalten werden, bis entweder eine Sekunde vergangen oder die Anzahl der Festschreibungsanforderungen 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 mit Hilfe von Tests mit mehreren Benutzern sicherstellen, dass die Erhöhung des Werts dieses Parameters die erwarteten Ergebnisse liefert.

Am Wert dieses Parameters vorgenommene Änderungen werden sofort wirksam. Sie brauchen nicht abzuwarten, bis alle Anwendungen von der Datenbank getrennt wurden.

Empfehlung: Erhöhen Sie den Wert dieses Parameters über seinen Standardwert hinaus, wenn mehrere Schreib-/Leseanwendungen regelmäßig gleichzeitig Datenbankfestschreibungen anfordern. Dadurch wird eine höhere Effizienz bei E/A-Operationen für Protokolldateien erzielt, da diese Operationen weniger häufig auftreten und bei jeder Operation mehr Protokollsätze geschrieben werden.

Sie könnten auch 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 würde der Systemaufwand für das 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:

  Mit Hilfe 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 (Protokollbereich für Arbeitseinheit)*

Zugehörige Referenzen:

- „uow_log_space_used - Unit of Work Log Space Used monitor element" in *System Monitor Guide and Reference*
- „commit_sql_stmts - Commit Statements Attempted monitor element" in *System Monitor Guide and Reference*
- „rollback_sql_stmts - Rollback Statements Attempted monitor element" in *System Monitor Guide and Reference*
- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

numarchretry - Anzahl der Wiederholungen bei Fehler

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]

Mit diesem Parameter wird die Anzahl der Versuche angegeben, die DB2 zur Archivierung einer Protokolldatei im primären oder sekundären Archivierungsverzeichnis unternehmen soll, bevor DB2 versucht, Protokolldateien im Verzeichnis des Übernahmesystems zu speichern. Dieser Parameter wird nur verwendet, wenn der Datenbankkonfigurationsparameter *failarchpath* definiert ist. Wenn *numarchretry* nicht definiert wird, setzt DB2 die Versuche zur Archivierung im primären oder sekundären Protokollpfad kontinuierlich fort.

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

softmax - Vor bedingtem Prüfpunkt zu schreibende Protokollsätze

Konfigurationstyp	Datenbank
Parametertyp	Konfigurierbar
Standardwert [Bereich]	100 [1 – 100 * *logprimary*]
Maßeinheit	Prozentsatz der Größe einer primären Protokolldatei

Dieser Parameter hat folgende Funktionen:

- Beeinflussen der Anzahl von Protokolldateien, die für die Wiederherstellung nach einem Systemabsturz (z. B. nach einem Stromausfall) erforderlich sind. Wenn beispielsweise der Standardwert 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 Wiederherstellung 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 Wiederherstellungsfenster, bereits auf Platte geschrieben sind.
- Festlegen der Frequenz der bedingten Prüfpunkte.

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 festgeschrieben, jedoch wurden die Daten im Pufferpool aktualisiert.
- Die Änderungen wurden festgeschrieben, jedoch noch nicht vom Pufferpool auf die Festplatte geschrieben.
- Die Änderungen wurden festgeschrieben und vom Pufferpool auf die Festplatte geschrieben.

Wenn eine Datenbank erneut gestartet wird, werden die Protokolldateien verwendet, um eine Wiederherstellung der Datenbank nach dem Systemabsturz auszuführen, die sicherstellt, dass die Datenbank in einem konsistenten Zustand verbleibt (d. h., alle festgeschriebenen Transaktionen werden in der Datenbank nachvollzogen, und keine der nicht festgeschriebenen Transaktionen werden in der Datenbank nachvollzogen).

Der Datenbankmanager verwendet eine Protokollsteuerdatei, um festzustellen, welche Datensätze aus der Protokolldatei in der Datenbank nachvollzogen werden müssen. Diese Protokollsteuerdatei wird 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 einem gewissen erhöhten Systemaufwand während des Neustarts der Datenbank.

Die Protokollsteuerdatei wird immer dann auf die Festplatte geschrieben, wenn eine Protokolldatei voll ist, und außerdem bei bedingten Prüfpunkten. Sie können diesen Konfigurationsparameter dazu verwenden, zusätzliche bedingte Prüfpunkte auszulösen.

Die Ablaufsteuerung für bedingte Prüfpunkte wird mit Hilfe 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 von der Protokollsteuerdatei 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 Wiederherstellungsprozess 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 Wiederherstellungsfenster 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 Wiederherstellung verarbeitet werden, verringern.

Beachten Sie jedoch, dass mehr Auslöser von Seitenlöschfunktionen und häufigere bedingte Prüfpunkte den Systemaufwand erhöhen, der 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 COMMIT-Punkten 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.

Zugehörige Referenzen:

- „logfilsiz - Protokolldateigröße" auf Seite 456
- „logprimary - Anzahl primärer Protokolldateien" auf Seite 458
- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

userexit - Benutzerexit aktivieren

Konfigurationstyp	Datenbank
Parametertyp	Konfigurierbar
Standardwert [Bereich]	No [Yes; No]

Wenn dieser Parameter aktiviert ist, wird unabhängig von der Einstellung des Parameters *logretain* eine Protokollierung mit Protokollspeicherung ausgeführt. Mit diesem Parameter wird außerdem angegeben, dass ein Benutzerexitprogramm verwendet werden soll, um Protokolldateien zu archivieren und wieder abzurufen. Protokolldateien werden archiviert, wenn sie vom Datenbankmanager geschlossen werden. Die Protokolldateien werden wieder abgerufen, wenn das Dienstprogramm ROLLFORWARD sie zur Wiederherstellung einer Datenbank benötigt.

Nach dem Aktivieren des Parameters *logretain* und/oder *userexit* muss eine Gesamtsicherung der Datenbank ausgeführt werden. Dieser Status wird durch den Markierungsparameter *backup_pending* angezeigt.

Wenn beide Parameter inaktiviert werden, ist die aktualisierende Wiederherstellung der Datenbank nicht mehr möglich, da keine Protokolldateien mehr gespeichert werden. In diesem Fall löscht der Datenbankmanager alle Protokolldateien im Verzeichnis *logpath* (einschließlich der Online-Archivprotokolldateien), ordnet neue aktive Protokolldateien zu und reaktiviert die Umlaufprotokollierung.

Zugehörige Referenzen:

- „Benutzerexit zur Datenbankwiederherstellung" in *Datenwiederherstellung und hohe Verfügbarkeit Handbuch und Referenz*
- „logretain - Beibehalten von Protokollen aktivieren" auf Seite 471
- „user_exit_status - Statusanzeiger für Benutzerexit" auf Seite 501
- „backup_pending - Sicherung anstehend" auf Seite 499
- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „ROLLFORWARD DATABASE Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

vendoropt - Lieferantenoptionen

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 [ ]

Mit diesem Parameter werden zusätzliche Parameter angegeben, die DB2 eventuell zur Kommunikation mit Speichersystemen bei Sicherungs-, Wiederherstellungs- oder Ladekopieoperationen benötigt.

Zugehörige Konzepte:

- Anhang E, „Knotenübergreifende Fehlerbehebung mit dem Befehl 'db2adutl' und den Datenbankkonfigurationsparametern 'logarchopt1' und 'vendoropt'", auf Seite 679

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*

- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

Wiederherstellung

Die folgenden Parameter wirken sich auf die verschiedenen Aspekte der Datenbankwiederherstellung aus:

- „autorestart - Automatischer Neustart aktiviert"
- „dft_loadrec_ses - Standardanzahl von Sitzungen für Wiederherstellung" auf Seite 478
- „indexrec - Zeitpunkt für Indexneuerstellung" auf Seite 483
- „num_db_backups - Anzahl der Datenbanksicherungen" auf Seite 484
- „rec_his_retentn - Aufbewahrungszeitraum für Wiederherstellungsprotokoll" auf Seite 485
- „trackmod - Geänderte Seiten protokollieren" auf Seite 486

Siehe auch „Wiederherstellung verteilter Arbeitseinheiten" auf Seite 488.

Die folgenden Parameter werden beim Einsatz von Tivoli Storage Manager (TSM) verwendet:

- „tsm_mgmtclass - Tivoli Storage Manager-Verwaltungsklasse" auf Seite 486
- „tsm_nodename - TSM-Knotenname" auf Seite 487
- „tsm_owner - TSM-Eignername" auf Seite 487
- „tsm_password - TSM-Kennwort" auf Seite 488

Die folgenden Parameter beziehen sich auf HADR (High Availability Disaster Recovery):

- „hadr_db_role - Rolle der HADR-Datenbank" auf Seite 479
- „hadr_local_host - Name des lokalen Hosts für HADR" auf Seite 479
- „hadr_local_svc - Lokaler HADR-Servicename" auf Seite 480
- „hadr_remote_host - Name des fernen Hosts für HADR" auf Seite 480
- „hadr_remote_inst - HADR-Exemplarname des fernen Servers" auf Seite 481
- „hadr_remote_svc - Ferner HADR-Servicename" auf Seite 481
- „hadr_syncmode - HADR-Synchronisationsmodus für Protokollierung im Peerstatus" auf Seite 481
- „hadr_timeout - HADR-Zeitlimitwert" auf Seite 482

autorestart - Automatischer Neustart aktiviert

Konfigurationstyp	Datenbank
Parametertyp	Online konfigurierbar
Weitergabeklasse	Sofort
Standardwert [Bereich]	On [On; Off]

Wenn dieser Parameter auf ON gesetzt wird, ruft der Datenbankmanager bei Bedarf automatisch das Dienstprogramm zum Neustarten der Datenbank auf, wenn eine Anwendung die Verbindung zur Datenbank herstellt. Das Programm zum Neustarten der Datenbank wird verwendet, um eine *Wiederherstellung nach Systemabsturz* auszuführen. Es wird ausgeführt, wenn die Datenbank abnormal beendet wurde, während Anwendungen mit ihr verbunden waren. Eine abnormale Beendigung der Datenbank könnte z. B. durch einen Stromausfall oder einen Feh-

ler der Systemsoftware verursacht werden. Bei der Wiederherstellung werden festgeschriebene Transaktionen, die sich im Pufferpool befanden, jedoch zum Zeitpunkt des Fehlers nicht auf die Festplatte geschrieben waren, in der Datenbank nachvollzogen. Außerdem werden alle nicht festgeschriebenen Transaktionen, die eventuell auf Platte geschrieben wurden, wieder zurückgesetzt.

Wenn der Parameter *autorestart* nicht aktiviert ist, empfängt eine Anwendung den Fehler SQL1015N, wenn sie versucht, die Verbindung zu einer Datenbank herzustellen, für die eine Wiederherstellung nach einem Systemabsturz ausgeführt werden muss (für die die Datenbank neu gestartet werden muss). In diesem Fall kann die Anwendung das Dienstprogramm zum Neustarten der Datenbank aufrufen, oder Sie können die Datenbank neu starten, indem Sie die Funktion zum Neustarten im Wiederherstellungsprogramm auswählen.

Zugehörige Konzepte:

- „Wiederherstellung nach einem Systemabsturz" in *Datenwiederherstellung und hohe Verfügbarkeit Handbuch und Referenz*

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESTART DATABASE Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

dft_loadrec_ses - Standardanzahl von Sitzungen für Wiederherstellung

Konfigurationstyp	Datenbank
Parametertyp	Online konfigurierbar
Weitergabeklasse	Sofort
Standardwert [Bereich]	1 [1 – 30 000]
Maßeinheit	Zähler

Mit diesem Parameter wird die Standardanzahl von Sitzungen angegeben, die während des Wiederabrufens einer Tabellenladekopie verwendet werden. Der Wert sollte auf eine optimale Anzahl von E/A-Sitzungen gesetzt werden, die zum Wiederabrufen einer Ladekopie verwendet werden sollen. Das Wiederabrufen einer Ladekopie ist eine ähnliche Operation wie die Wiederherstellung. 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 Wiederherstellung aktiviert ist.

Zugehörige Konzepte:

- „LOAD - Übersicht" in *Dienstprogramme für das Versetzen von Daten Handbuch und Referenz*

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*
- „Verschiedene Variablen" auf Seite 600

hadr_db_role - Rolle der HADR-Datenbank

Konfigurationstyp Datenbank

Gilt für

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

Parametertyp Informativ

Dieser Parameter gibt die aktuelle Rolle einer Datenbank an, unabhängig davon, ob sie online oder offline ist. Gültige Werte: STANDARD, PRIMARY oder STANDBY.

Anmerkung: Der Befehl GET SNAPSHOT FOR DATABASE liefert zwar den HADR-Status (High Availability Disaster Recovery), jedoch nur, wenn die Datenbank online ist.

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*

hadr_local_host - Name des lokalen Hosts für HADR

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 Null

Mit diesem Parameter wird der lokale Host für die HADR-TCP-Kommunikation (High Availability Disaster Recovery) angegeben. 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″.

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command″ in *Command Reference*
- „RESET DATABASE CONFIGURATION Command″ in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command″ in *Command Reference*

hadr_local_svc - Lokaler HADR-Servicename

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 Null

Mit diesem Parameter wird der TCP-Servicename oder die Portnummer angegeben, für die der HADR-Prozess (High Availability Disaster Recovery) Verbindungen akzeptiert.

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command″ in *Command Reference*
- „RESET DATABASE CONFIGURATION Command″ in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command″ in *Command Reference*

hadr_remote_host - Name des fernen Hosts für HADR

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 Null

Mit diesem Parameter wird der TCP/IP-Hostname oder die IP-Adresse des fernen HADR-Knotens (High Availability Disaster Recovery) angegeben. Ähnlich wie der Parameter *hadr_local_host* darf dieser Parameter nur einer IP-Adresse zugeordnet werden.

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command″ in *Command Reference*
- „RESET DATABASE CONFIGURATION Command″ in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command″ in *Command Reference*

hadr_remote_inst - HADR-Exemplarname des fernen Servers

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 Null

Mit diesem Parameter wird der Exemplarname des fernen Servers angegeben. Verwaltungstools wie die DB2-Steuerzentrale verwenden diesen Parameter, um eine Verbindung zu dem fernen Server herzustellen. Ebenso prüft HADR (High Availability Disaster Recovery), ob eine ferne Datenbank, die eine Verbindung anfordert, zu dem deklarierten fernen Exemplar gehört.

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

hadr_remote_svc - Ferner HADR-Servicename

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 Null

Mit diesem Parameter wird der TCP-Servicename oder die Portnummer angegeben, die vom fernen HADR-Knoten (High Availability Disaster Recovery) verwendet wird.

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

hadr_syncmode - HADR-Synchronisationsmodus für Protokollierung im Peerstatus

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] NEARSYNC [ASYNC; SYNC]

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. Gültige Werte:

SYNC Dieser Modus bietet den größten Schutz gegen Transaktionsverlust, jedoch auf Kosten einer längeren Transaktionsantwortzeit.

NEARSYNC Dieser Modus bietet einen etwas geringeren Schutz gegen Transaktionsverlust, jedoch im Gegenzug auch eine kürzere Transaktionsantwortzeit als der Modus SYNC.

ASYNC Dieser Modus hat das größte Risiko eines Transaktionsverlusts im Fall eines Ausfalls der Primärdatenbank, bietet allerdings auch die kürzeste Transaktionsantwortzeit unter den drei Modi.

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

hadr_timeout - HADR-Zeitlimitwert

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] 120 [1 – 4 294 967 295]

Dieser Parameter gibt die Zeit (in Sekunden) an, die der HADR-Prozess (High Availability Disaster Recovery) wartet, bevor ein Kommunikationsversuch als fehlgeschlagen eingestuft wird.

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

indexrec - Zeitpunkt für Indexneuerstellung

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; access]

Windows-Datenbankmanager
restart [restart; access]

Datenbank system (Systemwert verwenden) [system; restart; access]

Dieser Parameter gibt an, wann der Datenbankmanager versucht, ungültige Indizes neu zu erstellen. Für diesen Parameter sind drei Einstellungen möglich:

SYSTEM *Systemeinstellung verwenden* veranlasst, dass ungültige Indizes zu dem Zeitpunkt neu erstellt werden, der in der Konfigurationsdatei des Datenbankmanagers angegeben ist. (Anmerkung: Diese Einstellung ist nur für Datenbankkonfigurationen gültig.)

ACCESS *Bei Indexzugriff* veranlasst, dass ungültige Indizes neu erstellt werden, wenn zum ersten Mal auf den Index zugegriffen wird.

RESTART *Bei Datenbankneustart* veranlasst, dass ungültige Indizes neu erstellt werden, wenn der Befehl RESTART DATABASE explizit oder implizit abgesetzt wird. Beachten Sie, dass der Befehl RESTART DATABASE implizit abgesetzt wird, wenn der Parameter *autorestart* aktiviert ist.

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, besonders dann, 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 des Vorgangs, mit dem die Datenbank nach einem Systemabsturz wiederhergestellt und online verfügbar gemacht wird.

Bei der Einstellung „ACCESS" für diesen Parameter 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 wird, 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 Datenbankwiederherstellung werden alle ausführbaren Dateien von SQL-Prozeduren im Dateisystem entfernt, die zu der gerade wiederhergestellten Datenbank gehören. Wenn der Parameter *indexrec* auf RESTART gesetzt wird, 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 erst in das Dateisystem extrahiert, wenn diese SQL-Prozedur zum ersten Mal ausgeführt wird.

Zugehörige Tasks:

- „Backing up and restoring SQL procedures created prior to DB2 8.2" in *Application Development Guide: Building and Running Applications*

Zugehörige Referenzen:

- „autorestart - Automatischer Neustart aktiviert" auf Seite 477
- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESTART DATABASE Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

num_db_backups - Anzahl der Datenbanksicherungen

Konfigurationstyp	Datenbank
Parametertyp	Online konfigurierbar
Weitergabeklasse	Transaktionsgrenzwert
Standardwert [Bereich]	12 [1 — 32 768]

Dieser Parameter gibt die Anzahl der Datenbanksicherungen an, die für eine Datenbank beibehalten werden sollen. Wird die angegebene Anzahl Sicherungen erreicht, werden alte Sicherungen in der Datei des Wiederherstellungsprotokolls als abgelaufen markiert. Die Einträge in der Datei des Wiederherstellungsprotokolls

für Tabellenbereichssicherungen und Ladekopiesicherungen, die mit der abgelaufenen Datenbanksicherung verbunden sind, werden ebenfalls als abgelaufen markiert. Wird eine Sicherung als abgelaufen markiert, können die physischen Sicherungen von ihrem Speicherort (z. B. Platte, Band, TSM) entfernt werden. Die nächste Datenbanksicherung entfernt die abgelaufenen Einträge aus der Datei des Wiederherstellungsprotokolls.

Wird eine Datenbanksicherung in der Protokolldatei als abgelaufen markiert, werden entsprechende über einen DB2 Data Links Manager verbundene Dateisicherungen aus dem Archivierungsserver gelöscht.

Der Konfigurationsparameter *rec_his_retentn* sollte auf einen Wert gesetzt werden, der mit dem Wert von *num_db_backups* kompatibel ist. Wenn *num_db_backup* z. B. auf einen hohen Wert gesetzt ist, muss der Wert für *rec_his_retentn* hoch genug sein, um diese Anzahl der Sicherungen unterstützen zu können.

Zugehörige Referenzen:

- „rec_his_retentn - Aufbewahrungszeitraum für Wiederherstellungsprotokoll" auf Seite 485
- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

rec_his_retentn - Aufbewahrungszeitraum für Wiederherstellungsprotokoll

Konfigurationstyp	Datenbank
Parametertyp	Konfigurierbar
Standardwert [Bereich]	366 [-1; 0 — 30 000]
Maßeinheit	Tage

Mit diesem Parameter wird angegeben, wie viele Tage die Protokolldaten zu Sicherungen aufbewahrt werden sollen. Wenn die Datei des Wiederherstellungsprotokolls nicht benötigt wird, um Sicherungen, Wiederherstellungen und Ladevorgänge festzuhalten, kann dieser Parameter auf einen kleineren Wert gesetzt werden.

Wenn für diesen Parameter der Wert -1 angegeben wird, kann die Datei des Wiederherstellungsprotokolls nur explizit mit Hilfe der verfügbaren Befehle oder APIs entfernt werden. Wenn der Wert nicht -1 ist, wird die Datei des Wiederherstellungsprotokolls nach jeder vollständigen Sicherung der Datenbank entfernt.

Der Wert dieses Parameters überschreibt den Wert des Parameters *num_db_backups*, *rec_his_retentn* und *num_db_backups* müssen jedoch zusammenpassen. Wenn der Wert für *num_db_backups* groß ist, muss der Wert für *rec_his_retentn* groß genug sein, um die angegebene Anzahl von Sicherungen unterstützen zu können.

Unabhängig davon, wie kurz der Aufbewahrungszeitraum ist, werden die aktuelle Datenbanksicherung und die zugehörige Wiederherstellungsgruppe immer zurückbehalten, sofern Sie nicht das Dienstprogramm PRUNE mit der Angabe WITH FORCE OPTION verwenden.

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*
- „PRUNE HISTORY/LOGFILE Command" in *Command Reference*
- „num_db_backups - Anzahl der Datenbanksicherungen" auf Seite 484

trackmod - Geänderte Seiten protokollieren

Konfigurationstyp	Datenbank
Parametertyp	Konfigurierbar
Standardwert [Bereich]	No [Yes, No]

Wenn dieser Parameter auf "Yes" gesetzt wird, verfolgt der Datenbankmanager Datenbankänderungen, damit das Sicherungsprogramm feststellen kann, welche Untergruppen der Datenbankseiten bei einer Teilsicherung zu berücksichtigen und eventuell in das Sicherungsimage aufzunehmen sind. Wenn Sie diesen Parameter auf "Yes" setzen, müssen Sie zunächst eine vollständige Sicherung der Datenbank durchführen, damit Sie über eine Ausgangsbasis für Teilsicherungen verfügen. Wenn dieser Parameter aktiviert ist und ein Tabellenbereich erstellt wird, müssen Sie außerdem eine Sicherung erstellen, die den betreffenden Tabellenbereich umfasst. Dabei kann es sich entweder um eine Datenbanksicherung oder eine Tabellenbereichssicherung handeln. Nach erfolgter Sicherung dürfen Teilsicherungen diesen Tabellenbereich umfassen.

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

tsm_mgmtclass - Tivoli Storage Manager-Verwaltungsklasse

Konfigurationstyp	Datenbank
Parametertyp	Konfigurierbar
Standardwert [Bereich]	NULL [beliebige Zeichenfolge]

Die Tivoli Storage Manager-Verwaltungsklasse (TSM) legt fest, wie der TSM-Server die Sicherungsversionen der Objekte verwalten soll, die gesichert werden.

Standardmäßig ist für DB2 keine Verwaltungsklasse angegeben.

Beim Ausführen einer TSM-Sicherung versucht TSM vor Verwendung der im Datenbankkonfigurationsparameter angegebenen Verwaltungsklasse zunächst, das Sicherungsobjekt 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 Sicherungsobjekt erneut an die mit dem Datenbankkonfigurationsparameter angegebene Verwaltungsklasse.

Daher muss sowohl die Standardverwaltungsklasse als auch die mit dem Datenbankkonfigurationsparameter angegebene Verwaltungsklasse eine Sicherungskopiengruppe enthalten, da die Sicherungsoperation sonst fehlschlägt.

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*
- „Tivoli Storage Manager" in *Datenwiederherstellung und hohe Verfügbarkeit Handbuch und Referenz*

tsm_nodename - TSM-Knotenname

Konfigurationstyp Datenbank
Parametertyp Online konfigurierbar
Weitergabeklasse Anweisungsgrenzwert
Standardwert [Bereich] NULL [beliebige Zeichenfolge]

Mit diesem Parameter kann die Standardeinstellung für den Knotennamen, der dem Produkt Tivoli Storage Manager (TSM) zugeordnet ist, überschrieben werden. 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 die Sicherung ausgeführt wurde. Es ist möglich, dass der Wert für den Parameter *tsm_nodename* bei einer Sicherung mit DB2 (z. B. mit dem Befehl BACKUP DATABASE) überschrieben wird.

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

tsm_owner - TSM-Eignername

Konfigurationstyp Datenbank
Parametertyp Online konfigurierbar
Weitergabeklasse Anweisungsgrenzwert
Standardwert [Bereich] NULL [beliebige Zeichenfolge]

Mit diesem Parameter kann die Standardeinstellung für den Eigner, der dem Produkt Tivoli Storage Manager (TSM) zugeordnet ist, überschrieben werden. 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 einer Sicherung 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 die Sicherung ausgeführt wurde.

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

tsm_password - TSM-Kennwort

Konfigurationstyp Datenbank

Parametertyp Online konfigurierbar

Weitergabeklasse Anweisungsgrenzwert

Standardwert [Bereich] NULL [beliebige Zeichenfolge]

Mit diesem Parameter kann die Standardeinstellung für das Kennwort, das dem Produkt Tivoli Storage Manager (TSM) zugeordnet ist, überschrieben werden. 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 einer Sicherung 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 die Sicherung ausgeführt wurde. Es ist möglich, dass der Wert für den Parameter *tsm_nodename* bei einer Sicherung mit DB2 überschrieben wird.

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

Wiederherstellung verteilter Arbeitseinheiten

Die folgenden Parameter wirken sich auf die Wiederherstellung von DUOW-Transaktionen (Distributed Unit of Work - verteilte Arbeitseinheit) aus:

- „resync_interval - Intervall für Transaktionsresynchronisation"
- „spm_log_file_sz - Protokolldateigröße für SPM" auf Seite 489
- „spm_log_path - Protokolldateipfad für SPM" auf Seite 490
- „spm_max_resync - Maximale Anzahl von SPM-Resynchronisationsagenten" auf Seite 490
- „spm_name - Name des Synchronisationspunktmanagers" auf Seite 491
- „tm_database - Name für Transaktionsmanagerdatenbank" auf Seite 491

resync_interval - Intervall für Transaktionsresynchronisation

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

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 Umgebung mit verteilten Arbeitseinheiten (DUOW) ausführen. Dieser Parameter gilt ebenfalls für das Wiederherstellen von Systemen mit zusammengeschlossenen Datenbanken.

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.

Zugehörige Referenzen:

- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

spm_log_file_sz - Protokolldateigröße für SPM

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 — 1 000]
Maßeinheit	Seiten (4 KB)

Mit diesem Parameter wird die Größe der Protokolldatei für den Synchronisationspunktmanager (SPM) in 4-KB-Seiten angegeben. Die Protokolldatei befindet sich im Unterverzeichnis `spmlog` im Verzeichnis `sqllib` und wird erstellt, wenn der Synchronisationspunktmanager 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 COMMIT- oder ROLLBACK-Operationen 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.

Zugehörige Referenzen:

- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

spm_log_path - Protokolldateipfad für SPM

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] sqllib/spmlog [ein beliebiger gültiger Pfad oder eine beliebige gültige Einheit]

Dieser Parameter gibt an, in welches Verzeichnis die SPM-Protokolle geschrieben werden. Standardmäßig werden die Protokolle 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.

Zugehörige Referenzen:

- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

spm_max_resync - Maximale Anzahl von SPM-Resynchronisationsagenten

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]

Mit diesem Parameter wird die Anzahl der Agenten angegeben, die gleichzeitig Resynchronisationsoperationen ausführen können.

Zugehörige Referenzen:

- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

spm_name - Name des Synchronisationspunktmanagers

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

Mit diesem Parameter wird der Name des Exemplars des Synchronisationspunktmanagers (SPM) gegenüber dem Datenbankmanager identifiziert.

Zugehörige Referenzen:

- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

tm_database - Name für Transaktionsmanagerdatenbank

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]

Mit diesem Parameter wird der Name der Transaktionsmanagerdatenbank (TMD) für jedes DB2-Exemplar angegeben. Eine Transaktionsmanagerdatenbank kann eine der folgenden Datenbanken sein:

- Eine lokale Datenbank von DB2 Universal Database
- Eine ferne Datenbank von DB2 Universal Database, die sich nicht auf einem Host oder System IBM 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 Wiederherstellung 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 Exemplaren erstellen und diese Datenbanken ausschließlich als Transaktionsmanagerdatenbanken verwenden.

Zugehörige Referenzen:

- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

Datenbankverwaltung

Eine Reihe von Parametern stellt Informationen zur Datenbank bereit oder beeinflusst die Verwaltung Ihrer Datenbank. Diese werden in folgende Kategorien unterteilt:

- „Query Enabler"
- „Attribute" auf Seite 493
- „DB2 Data Links Manager" auf Seite 496
- „Status" auf Seite 499
- „Compilereinstellungen" auf Seite 502
- „Automatische Verwaltung" auf Seite 508

Query Enabler

Der folgende Parameter stellt Informationen für die Steuerung von Query Enabler bereit:

- „dyn_query_mgmt - Dynamische SQL-Abfrageverwaltung"

dyn_query_mgmt - Dynamische SQL-Abfrageverwaltung

Konfigurationstyp	Datenbank
Parametertyp	Online konfigurierbar
Standardwert [Bereich]	0 (DISABLE) [1(ENABLE), 0 (DISABLE)]

Dieser Parameter ist dort relevant, wo DB2 Query Patroller installiert ist. Wenn dieser Parameter auf den Wert „ENABLE" gesetzt wird, erfasst Query Patroller Informationen über die Abfrage, wie zum Beispiel die ID des übergebenden Benutzers und den geschätzten Ausführungsaufwand, der durch das Optimierungsprogramm berechnet wurde. Anhand dieser Werte wird unter Beachtung der Schwellenwerte auf Benutzer- und Systemebene bestimmt, ob die Abfrage durch Query Patroller verwaltet werden sollte.

Wenn dieser Parameter auf den Wert „DISABLE" gesetzt wird, erfasst Query Patroller keine Informationen zu übergebenen Abfragen, und es findet keine Abfrageverwaltung statt.

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

Attribute

Die folgenden Parameter stellen allgemeine Informationen zur Datenbank bereit:

- „alt_collate - Alternative Sortierfolge"
- „codepage - Codepage für die Datenbank" auf Seite 494
- „codeset - Codierter Zeichensatz für die Datenbank" auf Seite 494
- „collate_info - Informationen zur Sortierfolge" auf Seite 494
- „country - Datenbankgebietscode" auf Seite 495
- „database_level - Releasestand der Datenbank" auf Seite 495
- „release - Releasestand der Datenbankkonfiguration" auf Seite 495
- „territory - Datenbankgebiet" auf Seite 496

Mit Ausnahme des Parameters *alt_collate* dienen diese Parameter lediglich der Information.

alt_collate - Alternative Sortierfolge

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]

Mit diesem Parameter wird die Sortierfolge angegeben, die für Unicode-Tabellen in einer Nicht-Unicode-Datenbank zu verwenden ist. 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.

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

codepage - Codepage für die Datenbank

Konfigurationstyp Datenbank

Parametertyp Informativ

Dieser Parameter zeigt die Codepage an, die zur Erstellung der Datenbank verwendet wurde. Der Wert des Parameters *codepage* wird mit Hilfe des Werts für den Parameter *codeset* abgeleitet.

Zugehörige Referenzen:

- „codeset - Codierter Zeichensatz für die Datenbank" auf Seite 494
- „GET DATABASE CONFIGURATION Command" in *Command Reference*

codeset - Codierter Zeichensatz für die Datenbank

Konfigurationstyp Datenbank

Parametertyp Informativ

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.

Zugehörige Referenzen:

- „codepage - Codepage für die Datenbank" auf Seite 494
- „GET DATABASE CONFIGURATION Command" in *Command Reference*

collate_info - Informationen zur Sortierfolge

Dieser Parameter kann nur mit der API db2CfgGet angezeigt werden. Er kann **nicht** mit Hilfe des Befehlszeilenprozessors oder der Steuerzentrale 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. Sie können diese Byte wie einen Wert des Typs INTEGER (ganze Zahl) für die Plattform der Datenbank behandeln. Drei 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.

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.

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*

country - Datenbankgebietscode

Konfigurationstyp Datenbank

Parametertyp Informativ

Dieser Parameter zeigt den Gebietscode an, der beim Erstellen der Datenbank verwendet wurde.

Zugehörige Referenzen:

- „territory - Datenbankgebiet" auf Seite 496
- „GET DATABASE CONFIGURATION Command" in *Command Reference*

database_level - Releasestand der Datenbank

Konfigurationstyp Datenbank

Parametertyp Informativ

Dieser Parameter gibt den Releasestand des Datenbankmanagers an, der die Datenbank verwenden kann. Im Fall einer nicht abgeschlossenen oder fehlgeschlagenen Migration (Umstellung) spiegelt dieser Parameter den Releasestand der nicht umgestellten Datenbank wider und kann daher vom Parameter *release* (Releasestand der Datenbankkonfiguration) abweichen. Ansonsten ist der Wert des Parameters *database_level* mit dem Wert des Parameters *release* identisch.

Zugehörige Referenzen:

- „release - Releasestand der Datenbankkonfiguration" auf Seite 495
- „GET DATABASE CONFIGURATION Command" in *Command Reference*

release - Releasestand der Datenbankkonfiguration

Konfigurationstyp Datenbankmanager, Datenbank

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp Informativ

Dieser Parameter gibt den Releasestand der Konfigurationsdatei an.

Zugehörige Referenzen:

- „database_level - Releasestand der Datenbank" auf Seite 495
- „GET DATABASE CONFIGURATION Command" in *Command Reference*

- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

territory - Datenbankgebiet

Konfigurationstyp Datenbank

Parametertyp Informativ

Dieser Parameter zeigt das Gebiet an, mit dem die Datenbank erstellt wurde. Der Parameter *territory* wird vom Datenbankmanager zum Bestimmen der Werte für den Gebietscodeparameter (*territory*) verwendet.

Zugehörige Referenzen:

- „country - Datenbankgebietscode" auf Seite 495
- „GET DATABASE CONFIGURATION Command" in *Command Reference*

DB2 Data Links Manager

Die folgenden Parameter beziehen sich auf den DB2 Data Links Manager:

- „datalinks - Data Links-Unterstützung aktivieren"
- „dl_expint - Ablaufintervall für Data Links-Zugriffs-Token"
- „dl_num_copies - Anzahl Data Links-Kopien" auf Seite 497
- „dl_time_drop - Data Links-Zeit nach DROP" auf Seite 497
- „dl_token - Data Links-Tokenalgorithmus" auf Seite 498
- „dl_upper - Data Links-Token in Großbuchstaben" auf Seite 498
- „dl_wt_iexpint - Verfallsintervall für Data Links-Schreibtoken" auf Seite 499

datalinks - Data Links-Unterstützung aktivieren

Konfigurationstyp Datenbankmanager

Parametertyp Konfigurierbar

Standardwert [Bereich] NO [YES; NO]

Dieser Parameter gibt an, ob die Unterstützung für Data Links aktiviert ist. Der Wert „YES" gibt an, dass die Unterstützung für Data Links über den Data Links Manager aktiviert ist, der in Basisdateisystemen (z. B. JFS unter AIX) gespeicherte Dateien verbindet. Der Wert „NO" gibt an, dass die Unterstützung für Data Links nicht aktiviert ist.

Zugehörige Referenzen:

- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

dl_expint - Ablaufintervall für Data Links-Zugriffs-Token

Konfigurationstyp Datenbank

Parametertyp Online konfigurierbar

Weitergabeklasse Transaktionsgrenzwert

Standardwert [Bereich] 60 [1 — 31 536 000]

Maßeinheit Sekunden

Dieser Parameter gibt das Zeitintervall (in Sekunden) an, in dem das generierte Dateizugriffssteuerungs-Token gültig ist. Der Gültigkeitszeitraum beginnt mit dem Zeitpunkt der Generierung des Tokens. Der Data Links Filesystem Filter prüft die Gültigkeit des Tokens gegen dieses Gültigkeitsintervall.

Dieser Parameter gilt für DATALINK-Spalten, in denen „READ PERMISSION DB" angegeben ist.

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

dl_num_copies - Anzahl Data Links-Kopien

Konfigurationstyp Datenbank

Parametertyp Online konfigurierbar

Weitergabeklasse Sofort

Standardwert [Bereich] 0 [0 – 15]

Dieser Parameter gibt an, wie viele zusätzliche Kopien auf dem Archivierungsserver (z. B. auf einem TSM-Server) von einer Datei erstellt werden, wenn die Datei mit der Datenbank verbunden ist.

Der Standardwert für diesen Konfigurationsparameter ist Null (0).

Dieser Parameter gilt für DATALINK-Spalten, in denen „Recovery=Yes" angegeben ist.

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

dl_time_drop - Data Links-Zeit nach DROP

Konfigurationstyp Datenbank

Parametertyp Online konfigurierbar

Weitergabeklasse Sofort

Standardwert [Bereich] 1 [0 — 365]

Maßeinheit Tage

Dieser Parameter gibt den Zeitraum (in Tagen) an, in dem Dateien auf einem Archivierungsserver (z. B. auf einem TSM-Server) aufbewahrt werden, nachdem DROP DATABASE abgesetzt wurde.

Der Standardwert für diesen Konfigurationsparameter ist ein (1) Tag. Der Wert Null (0) bedeutet, dass die Dateien sofort nach dem Absetzen eines DROP-Befehls

vom Archivierungsserver gelöscht werden. (Die tatsächliche Datei wird nur gelöscht, wenn der Parameter ON UNLINK DELETE für die DATALINK-Spalte angegeben wurde.)

Dieser Parameter gilt für DATALINK-Spalten, in denen „Recovery=Yes" angegeben ist.

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

dl_token - Data Links-Tokenalgorithmus

Konfigurationstyp	Datenbank
Parametertyp	Online konfigurierbar
Weitergabeklasse	Transaktionsgrenzwert
Standardwert [Bereich]	MAC0 [MAC0; MAC1]

Dieser Parameter gibt den Algorithmus an, der bei der Generierung von DATALINK-Token für Dateizugriffssteuerung verwendet wird. Der Wert von MAC1 (Nachrichtenauthentifizierungscode) generiert einen sichereren Nachrichtenauthentifizierungscode als MAC0, der Leistungsaufwand ist jedoch höher.

Dieser Parameter gilt für DATALINK-Spalten, in denen „READ PERMISSION DB" angegeben ist.

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

dl_upper - Data Links-Token in Großbuchstaben

Konfigurationstyp	Datenbank
Parametertyp	Online konfigurierbar
Weitergabeklasse	Transaktionsgrenzwert
Standardwert [Bereich]	NO [YES; NO]

Der Parameter gibt an, ob die Token für Dateizugriffssteuerung Großbuchstaben verwenden. Der Wert „YES" gibt an, dass alle Zeichen in einem Token für die Zugriffssteuerung Großbuchstaben sind. Der Wert „NO" gibt an, dass das Token Groß- und Kleinbuchstaben enthalten kann.

Dieser Parameter gilt für DATALINK-Spalten, in denen „READ PERMISSION DB" angegeben ist.

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

dl_wt_iexpint - Verfallsintervall für Data Links-Schreibtoken

Konfigurationstyp	Datenbank
Parametertyp	Online konfigurierbar
Weitergabeklasse	Transaktionsgrenzwert
Standardwert [Bereich]	60 [1 – 31 536 000]
Maßeinheit	Sekunden

Dieser Parameter gibt das ursprüngliche Verfallsintervall eines Schreibtokens an, das aus einer DATALINK-Spalte generiert wurde. Das ursprüngliche Verfallsintervall ist der Zeitraum zwischen dem Generierungszeitpunkt des Tokens und dem Zeitpunkt, zu dem es zum ersten Mal zum Öffnen der Datei verwendet wird.

Dieser Parameter gilt nur für eine DATALINK-Spalte, die mit WRITE PERMISSION ADMIN definiert wurde.

Empfehlung: Der Wert sollte groß genug sein, um den Zeitraum zwischen dem Abrufen eines Schreibtokens und seiner Verwendung zum Öffnen der Datei abzudecken.

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

Status

Die folgenden Parameter stellen Informationen zum Status der Datenbank bereit:

- „backup_pending - Sicherung anstehend"
- „database_consistent - Datenbank ist konsistent" auf Seite 500
- „log_retain_status - Statusanzeiger für Beibehalten der Protokolle" auf Seite 500
- „multipage_alloc - Zuordnung aus mehreren Seiten bestehender Datei aktiv" auf Seite 500
- „restore_pending - Wiederherstellung anstehend" auf Seite 501
- „rollfwd_pending - Aktualisierende Wiederherstellung anstehend" auf Seite 501
- „user_exit_status - Statusanzeiger für Benutzerexit" auf Seite 501

backup_pending - Sicherung anstehend

Konfigurationstyp	Datenbank
Parametertyp	Informativ

Wenn dieser Parameter aktiviert ist, bedeutet dies, dass Sie eine Gesamtsicherung der Datenbank ausführen müssen, bevor Sie auf sie zugreifen. Dieser Parameter ist nur aktiviert, wenn die Datenbankkonfiguration geändert wurde, so dass die Datenbank jetzt nicht mehr nicht wiederherstellbar sondern wiederherstellbar ist (das bedeutet, die Parameter *logretain* und *userexit* waren zunächst auf NO gesetzt, später wird jedoch einer der Parameter (oder beide) auf YES gesetzt und die Aktualisierung der Datenbankkonfiguration wird akzeptiert).

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*

database_consistent - Datenbank ist konsistent

Konfigurationstyp Datenbank

Parametertyp Informativ

Dieser Parameter gibt an, ob die Datenbank in einem konsistenten Zustand ist.

YES gibt an, dass alle Transaktionen festgeschrieben oder rückgängig gemacht wurden, so dass 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, so dass 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.

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*

log_retain_status - Statusanzeiger für Beibehalten der Protokolle

Konfigurationstyp Datenbank

Parametertyp Informativ

Wenn dieser Parameter gesetzt ist, zeigt er an, dass Protokolldateien für die aktualisierende Wiederherstellung gespeichert werden.

Dieser Parameter wird gesetzt, wenn der Parameter *logretain* auf Recovery gesetzt ist.

Zugehörige Referenzen:

- „logretain - Beibehalten von Protokollen aktivieren" auf Seite 471
- „GET DATABASE CONFIGURATION Command" in *Command Reference*

multipage_alloc - Zuordnung aus mehreren Seiten bestehender Datei aktiv

Konfigurationstyp Datenbank

Parametertyp Informativ

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.

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. Wenn die Zuordnung mehrseitiger Dateien nicht erwünscht ist, muss die DB2-Registriervariable DB2_NO_MPFA_FOR_NEW_DB entsprechend definiert werden, bevor die Datenbank erstellt wird. Das Tool

db2empfa kann zur Aktivierung der Zuordnung mehrseitiger Dateien für eine Datenbank verwendet werden, für die dieses Merkmal inaktiviert ist.

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „Leistungsvariablen" auf Seite 586

restore_pending - Wiederherstellung anstehend

Konfigurationstyp Datenbank

Parametertyp Informativ

Dieser Parameter gibt an, ob sich die Datenbank im Status *Wiederherstellung anstehend* befindet.

Zugehörige Referenzen:

- „userexit - Benutzerexit aktivieren" auf Seite 475
- „GET DATABASE CONFIGURATION Command" in *Command Reference*

rollfwd_pending - Aktualisierende Wiederherstellung anstehend

Konfigurationstyp Datenbank

Parametertyp Informativ

Dieser Parameter kann einen der folgenden Status anzeigen:

- **DATABASE**, d. h. eine aktualisierende Wiederherstellung ist für diese Datenbank erforderlich
- **TABLESPACE**, d. h. mindestens ein Tabellenbereich muss aktualisierend wiederhergestellt werden
- **NO**, d. h. die Datenbank ist verwendbar und keine aktualisierende Wiederherstellung erforderlich

Die Wiederherstellung (mit ROLLFORWARD DATABASE) muss erfolgreich beendet sein, bevor auf die Datenbank bzw. den Tabellenbereich zugegriffen werden kann.

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*

user_exit_status - Statusanzeiger für Benutzerexit

Konfigurationstyp Datenbank

Parametertyp Informativ

Wenn der Wert dieses Parameters "Yes" ist, heißt dies, dass der Datenbankmanager für die aktualisierende Wiederherstellung aktiviert ist und dass das Benutzerexitprogramm verwendet wird, um Protokolldateien zu archivieren und wieder abzurufen, wenn es vom Datenbankmanager aufgerufen wird.

Zugehörige Referenzen:

- „userexit - Benutzerexit aktivieren" auf Seite 475
- „GET DATABASE CONFIGURATION Command" in *Command Reference*

Compilereinstellungen

Die folgenden Parameter stellen Informationen zur Verfügung, die den Compiler beeinflussen:

- „dft_degree - Standardgrad der Parallelität"
- „dft_mttb_types - Standardtypen von verwalteten Tabellen zur Optimierung" auf Seite 503
- „dft_queryopt - Standardabfrageoptimierungsklasse" auf Seite 503
- „dft_refresh_age - Standardaktualisierungsalter" auf Seite 504
- „dft_sqlmathwarn - Bei arithmetischen Ausnahmebedingungen fortsetzen" auf Seite 504
- „num_freqvalues - Anzahl der häufigsten Werte" auf Seite 506
- „num_quantiles - Anzahl der Quantile für Spalten" auf Seite 507

dft_degree - Standardgrad der Parallelität

Konfigurationstyp	Datenbank
Parametertyp	Online konfigurierbar
Weitergabeklasse	Sofort
Standardwert [Bereich]	1 [-1, 1 – 32 767]

Mit diesem Parameter wird der Standardwert für das Sonderregister CURRENT DEGREE und die Bindeoption DEGREE angegeben.

Der Standardwert ist 1.

Bei Angabe des Werts 1 wird keine partitionsinterne Parallelität verwendet. Der Wert -1 bedeutet, dass das Optimierungsprogramm den Grad der partitionsinternen Parallelität je nach Anzahl der Prozessoren und Art der Abfrage festlegt.

Der Grad der partitionsinternen Parallelität für eine SQL-Anweisung wird während der Kompilierung der Anweisung mit Hilfe 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

Zugehörige Referenzen:

- „max_querydegree - Maximaler Grad der Parallelität bei Abfragen" auf Seite 522
- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

dft_mttb_types - Standardtypen von verwalteten Tabellen zur Optimierung

Konfigurationstyp	Datenbank
Parametertyp	Konfigurierbar
Standardwert [Bereich]	SYSTEM [ALL, NONE, FEDERATED_TOOL, SYSTEM, USER oder eine Liste von Werten]

Mit diesem Parameter wird der Standardwert für das Sonderregister CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION angegeben. Der Wert dieses Registers bestimmt, welche Typen gespeicherter, mit REFRESH DEFERRED definierter Abfragetabellen der Abfrageoptimierung verwendet werden.

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.

Zugehörige Referenzen:

- „CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION special register" in *SQL Reference, Volume 1*

dft_queryopt - Standardabfrageoptimierungsklasse

Konfigurationstyp	Datenbank
Parametertyp	Online konfigurierbar
Weitergabeklasse	Sofort
Standardwert [Bereich]	5 [0 — 9]
Maßeinheit	Abfrageoptimierungsklasse (siehe unten)

Mit der Abfrageoptimierungsklasse können Sie das Optimierungsprogramm anweisen, beim Kompilieren von SQL-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 Bindebefehl verwendet werden.

Derzeit sind folgende Abfrageoptimierungsklassen definiert:

- 0 - Minimale Abfrageoptimierung
- 1 - Abfrageoptimierung 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

Zugehörige Referenzen:

- „SET CURRENT QUERY OPTIMIZATION statement" in *SQL Reference, Volume 2*
- „BIND Command" in *Command Reference*
- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*

- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*
- „LIST DRDA INDOUBT TRANSACTIONS Command" in *Command Reference*

dft_refresh_age - Standardaktualisierungsalter

Konfigurationstyp	Datenbank
Parametertyp	Konfigurierbar
Standardwert [Bereich]	0 [0, 99999999999999 (ANY)]

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. Diese Zeitdifferenz stellt die maximale Dauer seit der Verarbeitung einer Anweisung REFRESH TABLE für eine spezifische gespeicherte Abfragetabelle des Typs REFRESH DEFERRED dar, während der diese Übersichtstabelle zum Optimieren der Verarbeitung einer Abfrage verwendet werden kann. Wenn für CURRENT REFRESH AGE der Wert 99999999999999 (ANY) gilt und die Abfrageoptimierungsklasse (QUERY OPTIMIZATION) den Wert zwei (oder fünf und höher) hat, werden gespeicherte Abfragetabellen der Art REFRESH DEFERRED zum Optimieren der Verarbeitung einer dynamischen SQL-Abfrage herangezogen.

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

dft_sqlmathwarn - Bei arithmetischen Ausnahmebedingungen fortsetzen

Konfigurationstyp	Datenbank
Parametertyp	Konfigurierbar
Standardwert [Bereich]	No [No, Yes]

Mit diesem Parameter wird der Standardwert festgelegt, der bestimmt, ob arithmetische Fehler und Fehler bei Abfrageumwandlungen beim Kompilieren von SQL-Anweisungen als Fehler oder als Warnungen behandelt werden. 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 die Funktionsweise von Prüfungen auf Integritätsbedingung, Auslösern 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, Auslöser 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 DB2 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 auf „No" gesetzt wird, schlägt der Versuch, dieselbe Zeile einzufügen, fehl, weil der Fehler der Division durch 0 DB2 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. Zum 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.

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

num_freqvalues - Anzahl der häufigsten Werte

Konfigurationstyp	Datenbank
Parametertyp	Online konfigurierbar
Weitergabeklasse	Sofort
Standardwert [Bereich]	10 [0 — 32 767]
Maßeinheit	Zähler

Mit diesem Parameter können Sie die Anzahl der „häufigsten Werte" angeben, die gesammelt werden, wenn die Option WITH DISTRIBUTION im Befehl RUNSTATS angegeben wird. 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 SQL-Optimierungsprogramm mehr Daten zur Verfügung. Andererseits ist für den Katalog mehr Speicher 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 auch die Anzahl der häufigsten Werte angeben, die auf der Tabellen- oder Spaltenebene als Teil des Befehls RUNSTATS beibehalten werden. Wenn kein Wert angegeben wird, wird der Wert des Konfigurationsparameters *num_freqvalues* verwendet.

Durch Aktualisieren dieses Parameters kann das Optimierungsprogramm bessere Schätzwerte für die Auswahl von nicht gleichmäßig verteilten Daten mit Hilfe einiger Vergleichselemente (=, <, >, IS NULL, IS NOT NULL) erzielen. Exaktere Berechnungen der Selektivität können zur Auswahl effizienterer Zugriffspläne führen.

Nachdem Sie den Wert dieses Parameters verändert haben, müssen Sie wie folgt vorgehen:

- Führen Sie den Befehl RUNSTATS aus, nachdem alle Benutzer die Verbindung zur Datenbank getrennt und Sie die Verbindung zur Datenbank wiederhergestellt haben.
- Binden Sie alle Pakete mit statischen SQL-Anweisungen erneut.

Der Befehl RUNSTATS ermöglicht es über die Option NUM_REQVALUES, die Anzahl der häufigsten Werte anzugeben. Es ist einfacher, die Anzahl der häufigsten Werte über den Befehl RUNSTATS zu ändern, als über den Datenbankkonfigurationsparameter *num_freqvalues*.

Beim Verwenden des Befehls RUNSTATS haben Sie die Möglichkeit, die Anzahl der häufigsten Werte zu begrenzen, die auf Tabellenebene oder auf Spaltenebene gesammelt werden. 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: 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 mit Hilfe einer SQL-Anweisung SELECT geschehen, die eine Rangfolge des Vorkommens jedes Werts in einer Spalte liefert. Dabei dürfen Sie einheitlich verteilte, eindeutige, lange 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.

Zugehörige Referenzen:

- „num_quantiles - Anzahl der Quantile für Spalten" auf Seite 507
- „stat_heap_sz - Größe des Statistikzwischenspeichers" auf Seite 419
- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

num_quantiles - Anzahl der Quantile für Spalten

Konfigurationstyp	Datenbank
Parametertyp	Online konfigurierbar
Weitergabeklasse	Sofort
Standardwert [Bereich]	20 [0 – 32 767]
Maßeinheit	Zähler

Mit diesem Parameter wird die Anzahl der Quantile gesteuert, die gesammelt werden, wenn die Option WITH DISTRIBUTION im Befehl RUNSTATS angegeben wird. 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 SQL-Optimierungsprogramm mehr Daten zur Verfügung. Andererseits ist für den Katalog mehr Speicher erforderlich. Wenn die Werte 0 oder 1 angegeben werden, werden keine Quantil-Statistikdaten erhoben, auch wenn Sie anfordern, dass Verteilungsstatistikdaten gesammelt werden.

Sie können auch die Anzahl der Quantile angeben, die auf der Tabellen- oder Spaltenebene als Teil des Befehls RUNSTATS erfasst werden. Wenn kein Wert angegeben wird, wird der Wert des Konfigurationsparameters *num_quantiles* verwendet.

Durch Aktualisieren dieses Parameters können bessere Schätzwerte für die Auswahl von nicht gleichmäßig verteilten Daten mit Hilfe von Bereichsvergleichselementen erzielt werden. Unter anderem entscheidet das Optimierungsprogramm mit Hilfe 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.)

Nachdem Sie den Wert dieses Parameters verändert haben, müssen Sie wie folgt vorgehen:

- Führen Sie den Befehl RUNSTATS aus, nachdem alle Benutzer die Verbindung zur Datenbank getrennt und Sie die Verbindung zur Datenbank wiederhergestellt haben.

- Binden Sie alle Pakete mit statischen SQL-Anweisungen erneut.

Der Befehl RUNSTATS ermöglicht die Angabe der Anzahl der zu erfassenden Quantile über die Option NUM_QUANTILES. Es ist einfacher, die Anzahl der zu erfassenden Quantile über den Befehl RUNSTATS zu ändern, als über den Datenbankkonfigurationsparameter *num_quantiles*.

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 garantiert einen maximalen Schätzfehler von ungefähr 2,5 % für alle einseitigen Bereichsvergleichselemente (>, >=, < oder <=) und einen maximalen Fehler von 5 % für jedes BETWEEN-Vergleichselement. Nachfolgend eine einfache Möglichkeit, um die Anzahl der Quantile einzugrenzen:

- Bestimmen Sie den maximalen Fehler, der bei der Schätzung der Anzahl von Zeilen jeder Bereichsabfrage tolerierbar ist, als Prozentwert P.
- Die Anzahl der Quantile sollte ca. 100/P sein, wenn die meisten Ihrer Vergleichselemente BETWEEN-Vergleichselemente sind, und 50/P, wenn die meisten Ihrer Vergleichselemente andere Arten von Bereichsvergleichselementen (<, <=, > oder >=) sind.

Zum Beispiel ergeben 25 Quantile einen maximalen Schätzfehler von 4% bei BETWEEN-Vergleichselementen und 2% bei Vergleichselementen mit ">". Ein geeigneter praktischer Wert für diesen Parameter liegt im Bereich zwischen 10 und 50.

Zugehörige Referenzen:

- „num_freqvalues - Anzahl der häufigsten Werte" auf Seite 506
- „stat_heap_sz - Größe des Statistikzwischenspeichers" auf Seite 419
- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

Automatische Verwaltung

Mit den folgenden Parametern können die automatischen Verwaltungsaktivitäten verschiedener DB2-Dienstprogramme gesteuert werden:

- „autonomic_switches - Schalter für automatische Verwaltung"

autonomic_switches - Schalter für automatische Verwaltung

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	Alle Schalter inaktiviert

Über diesen Parameter können Sie eine Reihe von Schaltern einstellen, die intern jeweils als ein Bit des Parameters dargestellt werden. Sie können jeden dieser Schalter unabhängig aktualisieren, indem Sie die folgenden Parameter festlegen:

auto_maint	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_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 automatische Verwaltung global aktiviert bzw. inaktiviert werden.
auto_db_backup	Dieser Parameter der automatischen Verwaltung aktiviert bzw. inaktiviert automatische Sicherungsoperationen für eine Datenbank. Eine Sicherungsrichtlinie (eine definierte Gruppe von Regeln oder Bestimmungen) kann zur Angabe der automatischen Funktionsweise verwendet werden. Die Sicherungsrichtlinie hat den Zweck, sicherzustellen, dass die Datenbank regelmäßig gesichert wird. Die Sicherungsrichtlinie für eine Datenbank wird automatisch erstellt, wenn der DB2-Diagnosemonitor zum ersten Mal ausgeführt wird. Zur Aktivierung muss dieser Parameter auf den Wert 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.
auto_runstats	Dieser Parameter der automatischen Tabellenverwaltung aktiviert bzw. inaktiviert automatische RUNSTATS-Operationen für Tabellen in einer Datenbank. Eine Richtlinie (eine definierte Gruppe von Regeln oder Bestimmungen) für die Statistikerstellung 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 den Wert ON gesetzt werden und die übergeordneten Parameter müssen ebenfalls aktiviert sein.

auto_stats_prof
Dieser Parameter der automatischen Tabellenverwaltung aktiviert die Statistikprofilgenerierung, die dazu dient, Anwendungen zu unterstützen, deren Auslastungen komplexe Abfragen, zahlreiche Vergleichselemente, Verknüpfungen und Gruppierungen unter Einbeziehung verschiedener Tabellen beinhalten. Zur Aktivierung muss dieser Parameter auf den Wert ON gesetzt werden und die übergeordneten Parameter müssen ebenfalls aktiviert sein.

auto_prof_upd
Dieser Parameter der automatischen Tabellenverwaltung (ein untergeordneter Parameter von *auto_stats_prof*) gibt an, dass das RUNSTATS-Profil mit Empfehlungen zu aktualisieren ist. Wenn dieser Parameter inaktiviert wird, 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 den Wert ON gesetzt werden und die übergeordneten Parameter müssen ebenfalls aktiviert sein.

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 den Wert ON gesetzt werden und die übergeordneten Parameter müssen ebenfalls aktiviert sein.

Zugehörige Referenzen:

- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

Kommunikation

Die folgenden Gruppen von Parametern stellen Informationen zur Verwendung von DB2 in einer Client-/Serverumgebung bereit:

- „Konfiguration der Kommunikationsprotokolle"
- „DB2 Discovery" auf Seite 513

Konfiguration der Kommunikationsprotokolle

Die folgenden Parameter dienen zur Konfiguration Ihrer Datenbankclients und Datenbankserver:

- „nname - Name der NetBIOS-Workstation"
- „svcename - TCP/IP-Servicename"
- „tpname - APPC-Transaktionsprogrammname" auf Seite 512

nname - Name der NetBIOS-Workstation

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 diesem Parameter können Sie dem Datenbankexemplar auf einer Workstation in der NetBIOS-LAN-Umgebung einen eindeutigen Namen zuordnen. Der Wert von *nname* ist die Basis für die tatsächlichen NetBIOS-Namen, die in NetBIOS für eine Workstation registriert werden.

Da das NetBIOS-Protokoll seine Verbindungen mit Hilfe dieser NetBIOS-Namen erstellt, muss der Parameter *nname* sowohl für den Client als auch für den Server festgelegt werden.

Client-Anwendungen müssen den *nname*-Wert des Servers mit der Datenbank kennen, auf die zugegriffen werden soll. Der *nname*-Wert des Servers muss im Client-Knotenverzeichnis als Parameter „server-nname" mit dem Befehl CATALOG NETBIOS NODE katalogisiert werden.

Wenn der Knotenname *nname* auf dem Server-Knoten in einen neuen Namen geändert wird, müssen alle Clients, die auf Datenbanken dieses Servers zugreifen, diesen neuen Namen für den Server katalogisieren.

Zugehörige Referenzen:

- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „CATALOG NETBIOS NODE Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

svcename - TCP/IP-Servicename

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 Parameter enthält den Namen des TCP/IP-Ports, an dem ein Datenbankserver auf Datenübertragungen von fernen Clientknoten wartet. Dieser Name muss der erste zweier aufeinander folgender Ports sein, die für die Verwendung durch den Datenbankmanager reserviert sind. Der zweite Port wird zur Verarbeitung von Unterbrechungsanforderungen von Clients mit einer älteren Programmversion verwendet.

Damit Verbindungsanforderungen von einem Datenbankclient mit Hilfe 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. Wenn der Datenbankserver Anforderungen von Clients mit einer älteren Programmversion unterstützen soll, muss ein zweiter Port (Nummer *n+1* für Unterbrechungsanforderungen) in der Servicedatei auf dem Server definiert werden.

Der Port (Nummer *n*) des Datenbankservers und sein TCP/IP-Servicename müssen in der Servicedatei auf dem Datenbankclient definiert werden. Bei Clients mit einer älteren Programmversion muss außerdem der Unterbrechungsport (Nummer *n+1*) in der Servicedatei des Clients definiert werden.

Auf UNIX-basierten Systemen befindet sich die Servicedatei in folgendem Verzeichnis: `/etc/services`

Der Parameter *svcename* sollte auf den Servicenamen gesetzt werden, der dem Hauptverbindungsport zugeordnet ist, so dass der Datenbankserver nach dem Start ermitteln kann, an welchem Port er für eingehende Verbindungsanforderungen empfangsbereit sein muss. Wenn Sie einen Client mit einer älteren Programmversion unterstützen oder verwenden, wird der Servicename für den Unterbrechungsport nicht in der Konfigurationsdatei gespeichert. Die Nummer des Unterbrechungsports kann anhand der Nummer des Hauptverbindungsports ermittelt werden (*unterbrechungsportnummer* = *hauptverbindungsport* + 1).

Zugehörige Referenzen:

- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

tpname - APPC-Transaktionsprogrammname

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

Mit diesem Parameter wird der Name eines fernen Transaktionsprogramms definiert, das der Datenbankclient verwenden muss, wenn er unter Verwendung des APPC-Kommunikationsprotokolls eine Zuordnungsanforderung an den Datenbankserver absetzt. Dieser Parameter muss in der Konfigurationsdatei auf dem Datenbankserver festgelegt werden.

Der Wert dieses Parameters muss mit dem Transaktionsprogrammnamen übereinstimmen, der in der SNA-Transaktionsprogrammdefinition konfiguriert ist.

Empfehlung: Nur folgende Zeichen sind zur Verwendung in diesem Namen zulässig:

- Buchstaben (A - Z oder a - z)
- Numerische Zeichen (0 - 9)
- Dollarzeichen ($), Nummernzeichen (#), kommerzielles A (@) und Punkt (.)

Zugehörige Referenzen:

- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

DB2 Discovery

Die folgenden Parameter dienen zur Einrichtung von DB2 Discovery:

- „discover - Discovery-Modus"
- „discover_db - Discovery-Unterstützung für diese Datenbank" auf Seite 514
- „discover_inst - Discovery-Serverexemplar" auf Seite 514

discover - Discovery-Modus

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 für *discover* SEARCH angegeben ist, kann der Client Discovery-Anforderungen mit dem Parameter SEARCH zum Suchen von DB2-Serversystemen im Netzwerk absetzen. Discovery mit dem Parameter SEARCH stellt eine Implikation der durch Discovery mit dem Parameter KNOWN bereitgestellten Funktionalität zur Verfügung. Wenn für *discover* SEARCH 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 Exemplar- und Datenbankinformationen auf dem DB2-System an den Client zurückgegeben.
- Wenn für *discover* DISABLE angegeben ist, ist Discovery auf dem Client inaktiviert.

Der Standard-Discovery-Modus ist SEARCH.

Zugehörige Referenzen:
- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „Kommunikationsvariablen" auf Seite 573

discover_db - Discovery-Unterstützung für diese Datenbank

Konfigurationstyp	Datenbank
Parametertyp	Online konfigurierbar
Weitergabeklasse	Sofort
Standardwert [Bereich]	Enable [Disable, Enable]

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.

Standardmäßig ist dieser Parameter so eingestellt, dass 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.

Zugehörige Referenzen:
- „GET DATABASE CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE CONFIGURATION Command" in *Command Reference*

discover_inst - Discovery-Serverexemplar

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]

Dieser Parameter gibt an, ob dieses Exemplar von DB2 Discovery aufgespürt werden kann. Der Standardwert, ″enable″, gibt an, dass das Exemplar aufgespürt werden kann, der Wert ″disable″ dagegen verhindert, dass das Exemplar aufgespürt wird.

Zugehörige Referenzen:

- „GET DATABASE MANAGER CONFIGURATION Command″ in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command″ in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command″ in *Command Reference*

Umgebung mit partitionierter Datenbank

Die folgenden Gruppen von Parametern stellen Informationen zum Parallelbetrieb und zu Umgebungen mit partitionierten Datenbanken bereit:

- „Kommunikation″
- „Parallelverarbeitung″ auf Seite 522

Kommunikation

Die folgenden Parameter stellen Informationen zur Kommunikation in der Umgebung mit partitionierten Datenbanken bereit:

- „conn_elapse - Antwortzeit für Verbindung″
- „fcm_num_anchors - Anzahl von FCM-Nachrichtenankern″ auf Seite 516
- „fcm_num_buffers - Anzahl FCM-Puffer″ auf Seite 517
- „fcm_num_connect - Anzahl von FCM-Verbindungseinträgen″ auf Seite 518
- „fcm_num_rqb - Anzahl von FCM-Anforderungsblöcken″ auf Seite 519
- „max_connretries - Anzahl Wiederholungen für Verbindungsaufbau zum Knoten″ auf Seite 520
- „max_time_diff - Maximale Zeitdifferenz zwischen Knoten″ auf Seite 520
- „start_stop_time - Zeitlimit für DB2START und DB2STOP″ auf Seite 521

conn_elapse - Antwortzeit für Verbindung

Konfigurationstyp	Datenbankmanager
Gilt für	Partitionierten Datenbankserver mit lokalen und fernen Clients
Parametertyp	Online konfigurierbar
Weitergabeklasse	Sofort
Standardwert [Bereich]	10 [0–100]
Maßeinheit	Sekunden

Dieser Parameter gibt an, innerhalb wie viel Sekunden eine TCP/IP-Verbindung zwischen zwei Datenbankpartitionsservern aufgebaut werden muss. Wird der Verbindungsaufbau innerhalb der angegebenen Zeit erfolgreich durchgeführt, wird der Datenaustausch freigegeben. 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.

Zugehörige Referenzen:

- „max_connretries - Anzahl Wiederholungen für Verbindungsaufbau zum Knoten" auf Seite 520
- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

fcm_num_anchors - Anzahl von FCM-Nachrichtenankern

Konfigurationstyp	Datenbankmanager
Gilt für	• Datenbankserver mit lokalen und fernen Clients • Datenbankserver mit lokalen Clients • Partitionierten Datenbankserver mit lokalen und fernen Clients • Satellitendatenbankserver mit lokalen Clients
Parametertyp	Online konfigurierbar
Weitergabeklasse	Sofort
Standardwert [Bereich]	-1 [-1, 128 — *fcm_num_rqb*] In nicht partitionierten Datenbanksystemen muss der Parameter *intra_parallel* aktiviert werden, bevor der Parameter *fcm_num_anchors* verwendet werden kann.

Mit diesem Parameter wird die Anzahl von *FCM-Nachrichtenankern* angegeben. Agenten verwenden die Nachrichtenanker, um Nachrichten untereinander zu versenden. Der Standardwert (-1) gibt 75 % des Werts an, der für *fcm_num_rqb* angegeben ist.

Zugehörige Konzepte:

- „FCM-Kommunikation" in *Systemverwaltung: Implementierung*

Zugehörige Referenzen:

- „fcm_num_buffers - Anzahl FCM-Puffer" auf Seite 517
- „fcm_num_connect - Anzahl von FCM-Verbindungseinträgen" auf Seite 518
- „fcm_num_rqb - Anzahl von FCM-Anforderungsblöcken" auf Seite 519
- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

fcm_num_buffers - Anzahl FCM-Puffer

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]

32-Bit-Plattformen
512, 1 024 oder 4 096 [128 — *65 300*]

64-Bit-Plattformen
512, 1 024 oder 4 096 [128 — *524 288*]

- Datenbankserver mit lokalen und fernen Clients: Standardwert ist 1 024.
- Datenbankserver mit lokalen Clients: Standardwert ist 512.
- Partitionierter Datenbankserver mit lokalen und fernen Clients: Standardwert ist 4 096.

Auf Datenbanksystemen mit einer Partition wird dieser Parameter nur dann verwendet, wenn der Parameter *intra_parallel* aktiv ist.

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.

Wenn Sie über mehrere logische Knoten auf derselben Maschine verfügen, kann es erforderlich sein, den Wert für diesen Parameter zu erhöhen. Außerdem kann es erforderlich sein, den Wert für diesen Parameter zu erhöhen, wenn wegen der Anzahl von Benutzern im System, wegen der Anzahl von Datenbankpartitionsservern im System oder wegen der Komplexität der Anwendungen die verfügbaren Nachrichtenpuffer nicht ausreichen.

Wenn Sie auf Nicht-AIX-Systemen 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 benutzt. Unter AIX gilt dagegen Folgendes:

- Wenn in dem vom Datenbankmanager verwendeten allgemeinen Speicher genug Speicherplatz vorhanden ist, wird der FCM-Pufferzwischenspeicher von dort zugeordnet. In diesem Fall erhält jeder Datenbankpartitionsserver die im Parameter *fcm_num_buffers* angegebene Anzahl Puffer, d. h. es gibt keinen von allen Datenbankpartitionsservern gemeinsam benutzten Pool mit FCM-Puffern (dies war in DB2 Version 5 neu).
- Ist in dem vom Datenbankmanager verwendeten allgemeinen Speicher nicht genug Speicherplatz verfügbar, wird der FCM-Pufferzwischenspeicher aus einem separaten Speicherbereich (gemeinsamer AIX-Speicher) zugeordnet, der von allen logischen Knoten auf derselben Maschine gemeinsam benutzt wird. Ein Pool mit der im Parameter *fcm_num_buffers* angegebenen Anzahl von Puffern

wird von allen logischen Knoten auf derselben Maschine gemeinsam benutzt. Dies ist die Standardkonfiguration für alle Nicht-AIX-Plattformen.

Überprüfen Sie daher den verwendeten Wert. Überlegen Sie, wie viele FCM-Puffer auf der Maschine (oder den Maschinen) mit mehreren logischen Knoten insgesamt zugeordnet werden.

Zugehörige Konzepte:

- „FCM-Kommunikation" in *Systemverwaltung: Implementierung*

Zugehörige Referenzen:

- „fcm_num_connect - Anzahl von FCM-Verbindungseinträgen" auf Seite 518
- „fcm_num_anchors - Anzahl von FCM-Nachrichtenankern" auf Seite 516
- „fcm_num_rqb - Anzahl von FCM-Anforderungsblöcken" auf Seite 519
- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

fcm_num_connect - Anzahl von FCM-Verbindungseinträgen

Konfigurationstyp	Datenbankmanager
Gilt für	• Datenbankserver mit lokalen und fernen Clients • Datenbankserver mit lokalen Clients • Partitionierten Datenbankserver mit lokalen und fernen Clients • Satellitendatenbankserver mit lokalen Clients
Parametertyp	Online konfigurierbar
Weitergabeklasse	Sofort
Standardwert [Bereich]	-1 [-1, 128 — *fcm_num_rqb*] In nicht partitionierten Datenbanksystemen muss der Parameter *intra_parallel* aktiviert werden, bevor der Parameter *fcm_num_connect* verwendet werden kann.

Mit diesem Parameter wird die Anzahl von *FCM-Verbindungseinträgen* angegeben. Agenten verwenden die Verbindungseinträge, um Daten untereinander zu übergeben. Der Standardwert (-1) gibt 75 % des Werts an, der für *fcm_num_rqb* angegeben ist.

Zugehörige Konzepte:

- „FCM-Kommunikation" in *Systemverwaltung: Implementierung*

Zugehörige Referenzen:

- „fcm_num_buffers - Anzahl FCM-Puffer" auf Seite 517
- „fcm_num_anchors - Anzahl von FCM-Nachrichtenankern" auf Seite 516
- „fcm_num_rqb - Anzahl von FCM-Anforderungsblöcken" auf Seite 519

- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

fcm_num_rqb - Anzahl von FCM-Anforderungsblöcken

Konfigurationstyp Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients
- Satellitendatenbankserver mit lokalen Clients

Parametertyp Online konfigurierbar

Weitergabeklasse Sofort

Standardwert [Bereich]

UNIX-Plattformen (32 Bit)
256, 512, 2 048 [128 — 120 000]

UNIX-Plattformen (64 Bit)
256, 512, 2 048 [128 — 524 288]

Windows NT (32 Bit)
10 000 [250 — 2 097 152]

Windows NT (64 Bit)
256, 512, 2 048 [128 — 524 288]

- Für einen Datenbankserver mit lokalen und fernen Clients ist der Standardwert 512.
- Für einen Datenbankserver mit lokalen Clients ist der Standardwert 256.
- Für einen partitionierten Datenbankserver mit lokalen und fernen Clients ist der Standardwert 2 048.

In nicht partitionierten Datenbanksystemen muss der Parameter *intra_parallel* aktiviert werden, bevor der Parameter *fcm_num_rqb* verwendet werden kann.

Mit diesem Parameter wird die Anzahl von *FCM-Anforderungsblöcken* angegeben. Anforderungsblöcke sind das Medium, über das Informationen zwischen dem FCM-Dämon und einem Agenten bzw. zwischen Agenten übergeben werden.

Der Bedarf an Anforderungsblöcken variiert je nach Anzahl von Benutzern im System, Anzahl von Datenbankpartitionsservern im System und Komplexität von Abfragen. Beginnen Sie mit dem Standardwert und nutzen Sie bei der Optimierung dieses Parameters die Ergebnisse aus dem Datenbanksystemmonitor.

Zugehörige Konzepte:

- „FCM-Kommunikation" in *Systemverwaltung: Implementierung*

Zugehörige Referenzen:

- „fcm_num_buffers - Anzahl FCM-Puffer" auf Seite 517
- „fcm_num_connect - Anzahl von FCM-Verbindungseinträgen" auf Seite 518
- „fcm_num_anchors - Anzahl von FCM-Nachrichtenankern" auf Seite 516
- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

max_connretries - Anzahl Wiederholungen für Verbindungsaufbau zum Knoten

Konfigurationstyp	Datenbankmanager
Gilt für	Partitionierten Datenbankserver mit lokalen und fernen Clients
Parametertyp	Online konfigurierbar
Weitergabeklasse	Sofort
Standardwert [Bereich]	5 [0–100]

Wenn der Verbindungsaufbau zwischen zwei Datenbankpartitionsservern 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 Datenbankpartitionsserver herzustellen. Bei Überschreitung des für diesen Parameter angegebenen Werts wird eine Fehlernachricht zurückgegeben.

Zugehörige Referenzen:

- „conn_elapse - Antwortzeit für Verbindung" auf Seite 515
- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

max_time_diff - Maximale Zeitdifferenz zwischen Knoten

Konfigurationstyp	Datenbankmanager
Gilt für	Partitionierten Datenbankserver mit lokalen und fernen Clients
Parametertyp	Konfigurierbar
Standardwert [Bereich]	60 [1–1 440]
Maßeinheit	Minuten

Jeder Datenbankpartitionsserver verfügt über eine eigene Systemuhr. Dieser Parameter gibt die maximale Zeitdifferenz (in Minuten) an, die zwischen den in der Knotenkonfigurationsdatei aufgelisteten Datenbankpartitionsservern zulässig ist.

Sind einer Transaktion zwei oder mehr Datenbankpartitionsserver zugeordnet, deren Systemzeiten sich um mehr als diese Zeitdifferenz unterscheiden, wird die Transaktion zurückgewiesen und ein SQLCODE-Wert zurückgegeben. (Die Transaktion wird nur zurückgewiesen, wenn eine Datenänderung mit ihr verbunden ist.)

DB2 verwendet die *Weltzeit* (UTC - Coordinated Universal Time), damit unterschiedliche Zeitzonen beim Festlegen dieses Parameters nicht ins Gewicht fallen. Die Weltzeit ist identisch mit der Westeuropäischen Zeit (Greenwich Mean Time).

Zugehörige Referenzen:

- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

start_stop_time - Zeitlimit für DB2START und DB2STOP

Konfigurationstyp	Datenbankmanager
Gilt für	Datenbankserver mit lokalen und fernen Clients
Parametertyp	Online konfigurierbar
Weitergabeklasse	Sofort
Standardwert [Bereich]	10 [1 — 1 440]
Maßeinheit	Minuten

Mit diesem Parameter wird die Zeit (in Minuten) angegeben, innerhalb der alle Datenbankpartitionsserver auf den Befehl DB2START oder DB2STOP reagieren müssen. Außerdem wird er als Zeitlimit für ADD DBPARTITIONNUM-Operationen verwendet.

Datenbankpartitionsserver, die nicht innerhalb der angegebenen Zeit auf einen Befehl DB2START reagieren, senden eine Nachricht an das Fehlerprotokoll `db2start`, das sich im Unterverzeichnis `log` des Unterverzeichnisses `sqllib` im Ausgangsverzeichnis für das Exemplar befindet. Vor dem Neustart dieser Knoten sollten Sie auf diesen Knoten einen Befehl DB2STOP absetzen.

Datenbankpartitionsserver, die nicht innerhalb der angegebenen Zeit auf einen Befehl DB2STOP reagieren, senden eine Nachricht an das Fehlerprotokoll `db2stop`, das sich im Unterverzeichnis `log` des Unterverzeichnisses `sqllib` im Benutzerverzeichnis für das Exemplar befindet. Sie können DB2STOP entweder für jeden oder einmal für alle nicht reagierenden Datenbankpartitionsserver absetzen. (Die bereits gestoppten Server melden, dass sie bereits gestoppt sind.)

Zugehörige Referenzen:

- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „ADD DBPARTITIONNUM Command" in *Command Reference*

Parallelverarbeitung

Die folgenden Parameter stellen Informationen zur Parallelverarbeitung bereit:

- „intra_parallel - Partitionsinterne Parallelität aktivieren"
- „max_querydegree - Maximaler Grad der Parallelität bei Abfragen"

intra_parallel - Partitionsinterne Parallelität aktivieren

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.

Dieser Parameter gibt an, ob der Datenbankmanager partitionsinterne Parallelität verwenden kann.

Ist dieser Parameter auf "YES" gesetzt, können die durch die Parallelverarbeitung erreichten Leistungssteigerungen unter anderem bei Datenbankabfragen und der Indexerstellung genutzt werden.

Anmerkung: Wenn Sie diesen Parameterwert ändern, werden Pakete möglicherweise erneut an die Datenbank gebunden und es kann zu Leistungseinbußen kommen.

Zugehörige Referenzen:

- „max_querydegree - Maximaler Grad der Parallelität bei Abfragen" auf Seite 522
- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

max_querydegree - Maximaler Grad der Parallelität bei Abfragen

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 ermittelt)

Dieser Parameter gibt den maximalen Grad partitionsinterner Parallelität an, der für SQL-Anweisungen verwendet wird, die auf diesem Exemplar des Datenbankmanagers ausgeführt werden. Eine SQL-Anweisung verwendet bei ihrer Ausführung innerhalb einer Partition nicht mehr als diese Anzahl paralleler Operationen. Der Konfigurationsparameter *intra_parallel* muss aktiviert sein, damit die Datenbankpartition die partitionsinterne Parallelität verwenden kann.

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 mit Hilfe 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

Eine Ausnahmesituation bezüglich der Festlegung des tatsächlichen Parallelitätsgrads bei Abfragen tritt beim Erstellen eines Index auf. Ist in diesem Fall die Parallelität *intra_parallel* aktiviert und die Tabelle so groß, dass die Verwendung mehrerer Prozessoren vorteilhaft wäre, wird beim Erstellen eines Index die Anzahl der Online-Prozessoren (maximal jedoch 6) plus 1 Prozessor verwendet. Die Angabe des anderen oben genannten Parameters, der Bindeoption oder des Sonderregisters hat keine Auswirkung.

Zugehörige Referenzen:

- „intra_parallel - Partitionsinterne Parallelität aktivieren" auf Seite 522
- „dft_degree - Standardgrad der Parallelität" auf Seite 502
- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

Exemplarverwaltung

Eine Reihe von Parametern steht zur Verwaltung der Datenbankmanagerexemplare zur Verfügung. Diese Parameter werden in folgende Kategorien unterteilt:

- „Diagnose" auf Seite 524
- „Parameter für den Datenbanksystemmonitor" auf Seite 527
- „Systemverwaltung" auf Seite 529
- „Exemplarverwaltung" auf Seite 537

Diagnose

Mit den folgenden Parametern können die vom Datenbankmanager zur Verfügung gestellten Diagnoseinformationen gesteuert werden:

- „diaglevel - Aufzeichnungsebene bei Fehlerdiagnose"
- „diagpath - Verzeichnispfad für Diagnosedaten"
- „health_mon - Überwachung mit dem Diagnosemonitor" auf Seite 525
- „notifylevel - Benachrichtigungsstufe" auf Seite 526

diaglevel - Aufzeichnungsebene bei Fehlerdiagnose

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]

Durch diesen Parameter wird der Typ der Diagnosefehler festgelegt, die in der Datei 'db2diag.log' aufgezeichnet werden. Gültige Werte:

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 *diagpath* wird zur Angabe des Verzeichnisses verwendet, in dem sich die Fehlerdatei, die Ereignisprotokolldatei (nur unter Windows NT), die Alertprotokolldatei und alle Speicherauszugsdateien befinden, die je nach Wert des Parameters *diaglevel* generiert werden.

Empfehlung: Sie können den Wert dieses Parameters erhöhen, um zusätzliche Fehlerbestimmungsdaten zu sammeln, die zur Lösung eines Problems beitragen können.

Zugehörige Referenzen:

- „diagpath - Verzeichnispfad für Diagnosedaten" auf Seite 524
- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

diagpath - Verzeichnispfad für Diagnosedaten

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]

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.

Wenn dieser Parameter den Wert "Null" hat, werden die Diagnoseinformationen in Dateien eines der folgenden Verzeichnisse (bzw. Ordner) geschrieben:

- Für unterstützte Windows-Umgebungen:
  - Wenn die Umgebungsvariable bzw. das Schlüsselwort DB2INSTPROF **nicht** festgelegt ist, werden die Informationen in `x:\SQLLIB\DB2INSTANCE` geschrieben. Dabei ist `x:\SQLLIB` die Laufwerkangabe und das Verzeichnis, die in der Umgebungsvariablen bzw. dem Schlüsselwort DB2PATH angegeben sind, und `DB2INSTANCE` ist der Name des Exemplareigners.

    Anmerkung: Das Verzeichnis muss nicht SQLLIB heißen.
  - Wenn die Umgebungsvariable bzw. das Schlüsselwort DB2INSTPROF festgelegt ist, werden die Informationen in `x:\DB2INSTPROF\DB2INSTANCE` geschrieben. Dabei ist DB2INSTPROF der Name des Exemplarprofilverzeichnisses und DB2INSTANCE der Name des Exemplareigners.
- In auf UNIX basierenden Umgebungen: `INSTHOME/sqllib/db2dump`, wobei `INSTHOME` das Ausgangsverzeichnis (Home) des Exemplareigners ist.

Empfehlung: Verwenden Sie den Standardwert, oder geben Sie eine zentrale Speicherposition für die Diagnosedaten mehrerer Exemplare an.

In einer Umgebung mit partitionierten Datenbanken muss der angegebene Pfad auf einem gemeinsam benutzten Dateisystem angelegt sein.

Zugehörige Referenzen:

- „diaglevel - Aufzeichnungsebene bei Fehlerdiagnose" auf Seite 524
- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

health_mon - Überwachung mit dem Diagnosemonitor

Konfigurationstyp Datenbankmanager

Parametertyp Online konfigurierbar

Weitergabeklasse Sofort

Standardwert [Bereich] Off [On; Off]

Zugehörige Parameter

Mit diesem Parameter können Sie angeben, ob Sie anhand verschiedener Diagnoseanzeiger ein Exemplar, seine zugeordneten Datenbanken und Datenbankobjekte überwachen wollen. Wenn *health_mon* aktiviert ist, erfasst ein Agent Informationen zum ordnungsgemäßen Betrieb der ausgewählten Objekte. Wenn auf der Basis der von Ihnen festgelegten Schwellenwerte der Betrieb eines Objekts als nicht ordnungsgemäß eingestuft wird, können automatisch Benachrichtigungen versendet und Aktionen unternommen werden. Wenn *health_mon* inaktiviert ist (Standardeinstellung), wird der ordnungsgemäße Betrieb von Objekten nicht überwacht.

Sie können mit der Diagnosezentrale oder dem Befehlszeilenprozessor das Exemplar 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.

Zugehörige Referenzen:

- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

notifylevel - Benachrichtigungsstufe

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]

Dieser Parameter gibt den Typ der Hinweisnachrichten zur Systemverwaltung an, die in das Protokoll mit Benachrichtigungen für die Systemverwaltung geschrieben werden. Auf UNIX-Plattformen ist das Protokoll mit Benachrichtigungen für die Systemverwaltung eine Textdatei mit dem Namen *exemplar*.nfy. Unter Windows werden alle Hinweisnachrichten zur Systemverwaltung in das Ereignisprotokoll geschrieben. Die Fehler können von DB2, dem Diagnosemonitor, den Programmen Capture und Apply 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 Wiederherstellung nach einem dieser Fehler benötigen Sie gegebenenfalls 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 Wiederherstellung) protokolliert werden. Auf dieser Ebene werden Alarmnachrichten des Diagnosemonitors erfasst.

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.

4 — Informationsnachrichten.

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.

Empfehlung: Sie können den Wert dieses Parameters erhöhen, um zusätzliche Fehlerbestimmungsdaten zu sammeln, die zur Lösung eines Problems beitragen können. Wenn der Diagnosemonitor Benachrichtigung an die in seiner Konfiguration angegebenen Ansprechpartner senden soll, müssen Sie für den Parameter *notifylevel* einen Wert von 2 oder höher angeben.

Zugehörige Referenzen:

- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

Parameter für den Datenbanksystemmonitor

Mit den folgenden Parametern können verschiedene Funktionen des Datenbanksystemmonitors gesteuert werden:

- „dft_monswitches - Monitorschalter des Standarddatenbanksystems"

dft_monswitches - Monitorschalter des Standarddatenbanksystems

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	Alle Schalter ausgeschaltet außer dem standardmäßig eingeschalteten *dft_mon_timestamp*

Dieser Parameter ist insofern außergewöhnlich, als Sie mit ihm eine Reihe von Schaltern einstellen können, die intern jeweils als ein Bit des Parameters dargestellt werden. Sie können jeden dieser Schalter unabhängig aktualisieren, 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 den Systemaufwand 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.)

Zugehörige Referenzen:

- „GET MONITOR SWITCHES Command" in *Command Reference*
- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

Systemverwaltung

Die folgenden Parameter beziehen sich auf die Systemverwaltung:

- „comm_bandwidth - Kommunikationsbandbreite"
- „cpuspeed - CPU-Geschwindigkeit" auf Seite 530
- „dft_account_str - Standardzeichenfolge für Abrechnung" auf Seite 531
- „federated - Unterstützung für Systeme mit zusammengeschlossenen Datenbanken" auf Seite 532
- „jdk_path - Installationspfad für das Software Developer's Kit für Java" auf Seite 532
- „nodetype - Knotenart des Systems" auf Seite 533
- „numdb - Maximale Anzahl gleichzeitig aktiver Datenbanken einschließlich Host- und iSeries-Datenbanken" auf Seite 533
- „tp_mon_name - Name des Transaktionsprozessormonitors" auf Seite 534
- „util_impact_lim - Richtlinie für Exemplarauslastungswirkung" auf Seite 536

comm_bandwidth - Kommunikationsbandbreite

Konfigurationstyp	Datenbankmanager
Gilt für	Partitionierten Datenbankserver mit lokalen und fernen Clients
Parametertyp	Online konfigurierbar
Weitergabeklasse	Anweisungsgrenzwert
Standardwert [Bereich]	-1 [.1 – 100 000] Durch den Wert -1 wird der Parameter auf den Standardwert zurückgesetzt. Der Standardwert wird abhängig davon berechnet, ob ein Hochgeschwindigkeitsschalter verwendet wird.
Maßeinheit	Megabyte pro Sekunde

Der Wert, der (in MB pro Sekunde) für die Übertragungsbandbreite berechnet wird, wird vom SQL-Optimierungsprogramm 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 Anwendungen eventuell erneut binden (mit dem Befehl REBIND PACKAGE).

Zugehörige Referenzen:

- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

cpuspeed - CPU-Geschwindigkeit

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 [1^{-10} — 1] Der Wert -1 gibt an, dass der Wert dieses Parameters gemäß den Ergebnissen eines Messprogramms neu eingestellt wird.
Maßeinheit	Sekunden

Die CPU-Geschwindigkeit (in Millisekunden pro Instruktion) wird vom SQL-Optimierungsprogramm verwendet, um den Aufwand für die Ausführung bestimmter Operationen zu berechnen. Der Wert dieses Parameters wird automatisch bei der Installation des Datenbankmanagers auf der Grundlage der Ausgabe eines Programms zum Messen der CPU-Geschwindigkeit festgelegt. Dieses Programm wird ausgeführt, wenn Benchmark-Ergebnisse aus einem der folgenden Gründe nicht verfügbar sind:

- Auf der Plattform wird die Datei `db2spec.dat` nicht unterstützt.
- Die Datei `db2spec.dat` wird **nicht** gefunden.
- Die Daten für IBM RISC System/6000, Modell 530H werden in der Datei nicht gefunden.
- Die Daten für Ihr System werden in der Datei nicht gefunden.

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 Anwendungen eventuell erneut binden (mit dem Befehl REBIND PACKAGE).

Zugehörige Referenzen:

- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

dft_account_str - Standardzeichenfolge für Abrechnung

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 vom Anwendungsprogramm, das die API `sqlesact()` aufruft, oder dem Benutzer, der die Umgebungsvariable DB2ACCOUNT definiert, übergeben. 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 in der Abrechnungszeichenfolge folgende Zeichen:

- Alphabetische Zeichen (A - Z)
- Numerische Zeichen (0 - 9)
- Unterstreichungszeichen (_)

Zugehörige Referenzen:

- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

federated - Unterstützung für Systeme mit zusammengeschlossenen 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] No [Yes; No]

Dieser Parameter aktiviert bzw. inaktiviert Unterstützung für Anwendungen, die verteilte Anforderungen für von Datenquellen (wie die DB2-Produktfamilie und Oracle) verwaltete Daten übergeben.

Zugehörige Referenzen:

- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

jdk_path - Installationspfad für das Software Developer's Kit für Java

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]

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 berechnet.

Da es für diesen Parameter keinen Standardwert gibt, sollten Sie beim Installieren des SDK für Java einen Wert angeben.

Zugehörige Referenzen:

- „java_heap_sz - Maximale Zwischenspeichergröße für Java-Interpreter" auf Seite 429

- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

nodetype - Knotenart des Systems

Konfigurationstyp Datenbankmanager

Gilt für

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

Parametertyp Informativ

Dieser Parameter liefert Informationen zu den DB2-Produkten, die auf Ihrem System installiert sind, und daher auch zur Art Ihrer Datenbankmanagerkonfiguration. Die folgenden Werte können von diesem Parameter zurückgegeben werden, und dieser Knotenart können die folgenden Produkte zugeordnet sein:

- **Datenbankserver mit lokalen und fernen Clients:** Ein DB2-Serverprodukt, das lokale und ferne Datenbankclients unterstützt und auf andere ferne Datenbankserver zugreifen kann.
- **Client:** Ein Datenbankclient, der auf ferne Datenbankserver zugreifen kann.
- **Datenbankserver mit lokalen Clients:** Ein Verwaltungssystem für relationale DB2-Datenbanken, das lokale Datenbankclients unterstützt und auf andere ferne Datenbankserver zugreifen kann.
- **Partitionierter Datenbankserver mit lokalen und fernen Clients:** Ein DB2-Serverprodukt, das lokale und ferne Datenbankclients unterstützt, auf andere ferne Server zugreifen kann und mit Partitionsparallelität arbeiten kann.

Zugehörige Referenzen:

- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

numdb - Maximale Anzahl gleichzeitig aktiver Datenbanken einschließlich Host- und iSeries-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]

UNIX 8 [1 — 256]

Windows-Datenbankserver mit lokalen und fernen Clients 8 [1 — 256]

Windows-Datenbankserver mit lokalen Clients 3 [1 — 256]

Maßeinheit Zähler

Mit diesem Parameter wird die Anzahl der lokalen Datenbanken, die gleichzeitig aktiv sein können (d. h. auf die von Anwendungen zugegriffen werden kann), oder die maximale Anzahl verschiedener Datenbankaliasnamen angegeben, die in einem DB2 Connect-Server katalogisiert werden können. Jede Datenbank belegt Speicher, und eine aktive Datenbank verwendet ein neues Segment des gemeinsam benutzten Speichers.

Empfehlung: In der Regel ist es am besten, diesen Wert auf die tatsächliche Anzahl der bereits für den Datenbankmanager definierten Datenbanken zu setzen und um ca. 10 % für mögliches Wachstum zu erhöhen.

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. Beim Aktualisieren dieses Parameters sollten Sie die anderen Konfigurationsparameter beachten, die Speicher für eine Datenbank oder für eine mit der Datenbank verbundenen Anwendung zuordnen können.

Zugehörige Referenzen:

- „app_ctl_heap_sz - Zwischenspeichergröße für Anwendungssteuerung" auf Seite 406
- „sortheap - Zwischenspeichergröße für Sortierlisten" auf Seite 418
- „aslheapsz - Zwischenspeichergröße für Anwendungsunterstützungsebene" auf Seite 421
- „applheapsz - Zwischenspeichergröße für Anwendungen" auf Seite 412
- „locklist - Maximaler Speicher für Sperrenliste" auf Seite 399
- „dbheap - Zwischenspeicher für Datenbank" auf Seite 399
- „stmtheap - Größe des Anweisungszwischenspeichers" auf Seite 420
- „mon_heap_sz - Zwischenspeichergröße für Datenbanksystemmonitor" auf Seite 430
- „stat_heap_sz - Größe des Statistikzwischenspeichers" auf Seite 419
- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „database_memory - Größe des gemeinsam benutzten Datenbankspeichers" auf Seite 398

tp_mon_name - Name des Transaktionsprozessormonitors

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
- ENCINA
- CB
- SF
- TUXEDO
- TOPEND
- Kein Eintrag oder ein weiterer Wert (unter UNIX und Windows; keine weiteren gültigen Werte unter Solaris oder SINIX)

Mit diesem Parameter wird der Name des verwendeten Monitors für das Transaktionsprogramm (TP) angegeben.

- Sind Anwendungen in einer WebSphere Enterprise Edition CICS-Umgebung aktiv, muss für den Parameter „CICS" angegeben werden.
- Sind Anwendungen in einer WebSphere Enterprise Edition Encina-Umgebung aktiv, muss für den Parameter „ENCINA" angegeben werden.
- Sind Anwendungen in einer WebSphere Enterprise 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 DB2 Universal Database unter Windows NT 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 die Umgebungen UNIX und Windows NT.

Benutzer von TXSeries CICS und Encina: In vorangegangenen Versionen dieses Produkts unter Windows NT 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" oder „ENCINA" konfiguriert wird, ist dies ausreichend.

Benutzer von MQSeries: In vorangegangenen Versionen dieses Produkts unter Windows NT 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 NT 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 NT 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 DB2 Universal Database konfiguriert werden. Wenn mehrere TP-Monitore in einem einzigen DB2-Exemplar verwendet werden, muss diese Funktion verwendet werden.

Zugehörige Referenzen:

- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „xa_open-Zeichenfolgeformate" in *Systemverwaltung: Konzept*

util_impact_lim - Richtlinie für Exemplarauslastungswirkung

Konfigurationstyp	Datenbankmanager
Gilt für	• Datenbankserver mit lokalen Clients • Datenbankserver mit lokalen und fernen Clients • Partitionierten Datenbankserver mit lokalen und fernen Clients
Parametertyp	Online konfigurierbar
Standardwert [Bereich]	100 [1 - 100]
Maßeinheit	Prozentsatz der zulässigen Wirkung auf die Auslastung

Dieser Parameter ermöglicht dem Datenbankadministrator (DBA), die Leistungseinbußen für eine Auslastung durch ein gedrosseltes Dienstprogramm zu begrenzen. Der Datenbankadministrator kann so Onlinedienstprogramme während kritischer Auslastungszeiten des Produktionssystems ausführen und 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 (Standardwert) 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 wird, 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 (zum Beispiel 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 die Drosselung völlig, indem Sie den Parameter *util_impact_lim* auf den Wert 100 setzen.

Zugehörige Referenzen:

- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

Exemplarverwaltung

Die folgenden Parameter beziehen sich auf die Sicherheit und Verwaltung des Datenbankmanagerexemplars:

- „authentication - Authentifizierungstyp" auf Seite 538
- „catalog_noauth - Katalogisieren ohne Berechtigung zulässig" auf Seite 539
- „clnt_krb_plugin - Client-Kerberos-Plug-in" auf Seite 539
- „clnt_pw_plugin - Client-Plug-in für Benutzer-ID und Kennwort" auf Seite 540
- „dftdbpath - Standarddatenbankpfad" auf Seite 540
- „fed_noauth - Authentifizierung bei Servern mit zusammengeschlossenen Datenbanken umgehen" auf Seite 542
- „group_plugin - Gruppen-Plug-in" auf Seite 542
- „local_gssplugin - GSS-API-Plug-in für lokale Berechtigung auf Exemplarebene" auf Seite 543
- „srvcon_auth - Authentifizierungstyp für ankommende Verbindungen auf dem Server" auf Seite 543
- „srvcon_gssplugin_list - Liste der GSS-API-Plug-ins für ankommende Verbindungen auf dem Server" auf Seite 544
- „srvcon_pw_plugin - Plug-in für Benutzer-ID/Kennwort für ankommende Verbindungen auf dem Server" auf Seite 545
- „srv_plugin_mode - Server-Plug-in-Modus" auf Seite 545
- „sysadm_group - SYSADM-Gruppenname" auf Seite 546

- „sysctrl_group - SYSCTRL-Gruppenname" auf Seite 547
- „sysmaint_group - SYSMAINT-Gruppenname" auf Seite 547
- „sysmon_group - Berechtigungsgruppenname für Systemmonitor" auf Seite 548
- „trust_allclnts - Alle Clients akzeptieren" auf Seite 549
- „trust_clntauth - Authentifizierung gesicherter Clients" auf Seite 550
- „use_sna_auth - SNA-Authentifizierung verwenden" auf Seite 551

authentication - Authentifizierungstyp

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; KERBEROS; KRB_SERVER_ENCRYPT; GSSPLUGIN; GSS_SERVER_ENCRYPT]

Mit diesem Parameter wird angegeben und festgelegt, wie und wo die Authentifizierung eines Benutzers stattfindet.

Wird als Wert SERVER angegeben, werden die Benutzer-ID und das Kennwort vom Client an den Server gesendet, so dass die Authentifizierung auf dem Server ausgeführt werden kann. Der Wert SERVER_ENCRYPT unterscheidet sich vom Wert SERVER nur darin, dass über das Netzwerk gesendete Kennwörter verschlüsselt werden.

Anmerkung: Für eine Konfiguration, die der Common Criteria-Zertifizierung entspricht, ist SERVER der einzig 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 mit Hilfe 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) geeignet.

Zugehörige Referenzen:

- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

catalog_noauth - Katalogisieren ohne Berechtigung zulässig

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)]

Dieser Parameter gibt an, ob Benutzer Datenbanken und Knoten oder DCS- und ODBC-Verzeichnisse ohne SYSADM-Berechtigung katalogisieren und aus dem Katalog entfernen können. 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.

Zugehörige Referenzen:

- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

clnt_krb_plugin - Client-Kerberos-Plug-in

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 oder IBMkrb5 [beliebige gültige Zeichenfolge]

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. Standardmäßig ist der Wert null auf UNIX-basierten Systemen und `IBMkrb5` unter Windows-Betriebssystemen. Dieses Plug-in wird verwendet, wenn der Client mit der KERBEROS-Authentifizierung authentifiziert wird.

Zugehörige Referenzen:

- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

clnt_pw_plugin - Client-Plug-in für Benutzer-ID und Kennwort

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]

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. 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, SERVER oder SERVER_ENCRYPT arbeitet.

Zugehörige Referenzen:

- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

dftdbpath - Standarddatenbankpfad

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 Exemplareigners [beliebiger vorhandener Pfad]
Windows	Laufwerk, auf dem DB2 installiert ist [beliebiger vorhandener Pfad]

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.

In einer Umgebung mit partitionierten Datenbanken müssen Sie sicherstellen, dass der Pfad, in dem die Datenbank erstellt wird, kein an das Dateisystem NFS angehängter Pfad ist (auf UNIX-Plattformen) bzw. kein Netzwerklaufwerk 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 Knotennamen 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 sowohl aus Integritätsgründen als auch zur leichteren Sicherung und Wiederherstellung zu trennen.

In UNIX-basierten 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.

Zugehörige Referenzen:

- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

fed_noauth - Authentifizierung bei Servern mit zusammengeschlossenen Datenbanken umgehen

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 auf dem Exemplar umgangen. Es wird dann angenommen, dass die Authentifizierung an der Datenquelle vorgenommen wird. 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 für den Server mit zusammengeschlossenen Datenbanken SYSADM-Berechtigung erlangen.

Zugehörige Referenzen:

- „authentication - Authentifizierungstyp" auf Seite 538
- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „federated - Unterstützung für Systeme mit zusammengeschlossenen Datenbanken" auf Seite 532

group_plugin - Gruppen-Plug-in

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]

Mit diesem Parameter wird der Name der Gruppen-Plug-in-Bibliothek angegeben. Standardmäßig ist dieser Wert null und DB2 verwendet die Gruppensuchfunktion (Lookup) des Betriebssystems. Das Plug-in wird für alle Gruppensuchen verwendet.

Zugehörige Referenzen:

- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

local_gssplugin - GSS-API-Plug-in für lokale Berechtigung auf Exemplarebene

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]

Mit diesem Parameter wird der Name der GSS-API-Plug-in-Standardbibliothek angegeben, die zur lokalen Berechtigung auf Exemplarebene zu verwenden ist, wenn der Konfigurationsparameter *authentication* des Datenbankmanagers auf den Wert GSSPLUGIN oder GSS_SERVER_ENCRYPT gesetzt ist.

Zugehörige Referenzen:

- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

srvcon_auth - Authentifizierungstyp für ankommende Verbindungen auf dem Server

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 [CLIENT; SERVER; SERVER_ENCRYPT; KERBEROS; KRB_SERVER_ENCRYPT; GSSPLUGIN; GSS_SERVER_ENCRYPT]

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. 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 538 .

Zugehörige Referenzen:

- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

srvcon_gssplugin_list - Liste der GSS-API-Plug-ins für ankommende Verbindungen auf dem Server

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]

Mit diesem Parameter werden die GSS-API-Plug-in-Bibliotheken angegeben, die vom Datenbankserver unterstützt werden. 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. Dieser Parameter verarbeitet ankommende Verbindungen auf dem Server, wenn der Parameter *srvcon_auth* mit dem Wert KERBEROS, KRB_SERVER_ENCRYPT, GSSPLUGIN oder GSS_SERVER_ENCRYPT angegeben ist oder wenn *srvcon_auth* nicht angegeben ist und *authentication* mit dem Wert KERBEROS, KRB_SERVER_ENCRYPT, GSSPLUGIN oder GSS_SERVER_ENCRYPT angegeben ist.

Zugehörige Referenzen:

- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

srvcon_pw_plugin - Plug-in für Benutzer-ID/Kennwort für ankommende Verbindungen auf dem Server

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]

Mit diesem Parameter wird der Name der Benutzer-ID/Kennwort-Plug-in-Standardbibliothek angegeben, die für die serverseitige Authentifizierung zu verwenden ist. Standardmäßig ist dieser Wert null und die von DB2 bereitgestellte Benutzer-ID/Kennwort-Plug-in-Bibliothek wird verwendet.

Der Parameter verarbeitet ankommende Verbindungen auf dem Server, wenn der Parameter *srvcon_auth* mit dem Wert SERVER oder SERVER_ENCRYPT angegeben ist oder wenn *srvcon_auth* nicht angegeben ist und *authentication* mit dem Wert CLIENT, SERVER oder SERVER_ENCRYPT angegeben ist.

Zugehörige Referenzen:

- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

srv_plugin_mode - Server-Plug-in-Modus

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

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.

Zugehörige Referenzen:

- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

sysadm_group - SYSADM-Gruppenname

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 innerhalb des Datenbankmanagers dar, von der aus sämtliche Datenbankobjekte gesteuert werden. Mit diesem Parameter wird der Gruppenname definiert, der die Berechtigung SYSADM für das Datenbankmanagerexemplar besitzt.

Die Berechtigung SYSADM wird von den Sicherheitseinrichtungen festgelegt, die in einer bestimmten Betriebsumgebung verwendet werden.

- Unter Windows 98 muss für die SYSADM-Gruppe der Wert ″NULL″ angegeben werden.

  Dieser Parameter muss für Clients unter Windows 98 bei Verwendung von Systemschutz den Wert „NULL" haben, weil unter dem Betriebssystem Windows 98 keine Gruppeninformationen gespeichert werden und somit keine Möglichkeit zur Überprüfung besteht, ob ein Benutzer einer bestimmten SYSADM-Gruppe angehört. Bei Angabe eines Gruppennamens kann keiner der Benutzer dieser Gruppe angehören.
- Unter den Betriebssystemen Windows NT und Windows 2000 kann dieser Parameter auf jede lokale Gruppe gesetzt werden, deren Name aus 8 oder weniger Zeichen besteht und in der Sicherheitsdatenbank von Windows NT und Windows 2000 definiert ist. Wenn für diesen Parameter der Wert „NULL" definiert wird, haben alle Mitglieder der Administratorengruppe die Berechtigung SYSADM.
- Wenn bei auf UNIX basierenden Systemen der Wert „NULL" für diesen Parameter angegeben wird, gilt die Primärgruppe des Exemplareigners 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. Sie können auch das Notizbuch **Exemplar konfigurieren** über die DB2-Steuerzentrale aufrufen und den Parameter über das Notizbuch definieren.

Zugehörige Referenzen:

- „sysctrl_group - SYSCTRL-Gruppenname" auf Seite 547
- „sysmaint_group - SYSMAINT-Gruppenname" auf Seite 547
- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

sysctrl_group - SYSCTRL-Gruppenname

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 diesem Parameter wird der Gruppenname definiert, der die Berechtigung SYSCTRL (Systemsteuerung) besitzt. Die Berechtigung SYSCTRL umfasst Zugriffsrechte, die sich auf Systemressourcen auswirkende Operationen ermöglichen, aber keinen direkten Zugriff auf Daten zulassen.

Achtung: Dieser Parameter muss für Clients unter Windows 98 den Wert NULL besitzen, wenn Systemschutz verwendet wird (d. h., die Authentifizierung ist CLIENT, SERVER, DCS oder eine andere gültige Authentifizierung). Der Grund hierfür ist, dass Windows 98 keine Gruppeninformationen speichert und somit keine Möglichkeit bietet, festzustellen, ob ein Benutzer ein Mitglied einer bestimmten SYSCTRL-Gruppe ist. Bei Angabe eines Gruppennamens kann keiner der Benutzer dieser Gruppe angehören.

Wenn Sie den Parameter auf seinen Standardwert (NULL) zurücksetzen wollen, führen Sie den Befehl UPDATE DBM CFG USING SYSCTRL_GROUP NULL aus. Das Schlüsselwort „NULL" muss in Großbuchstaben angegeben werden. Sie können auch das Notizbuch **Exemplar konfigurieren** über die DB2-Steuerzentrale aufrufen und den Parameter über das Notizbuch definieren.

Zugehörige Referenzen:

- „sysadm_group - SYSADM-Gruppenname" auf Seite 546
- „sysmaint_group - SYSMAINT-Gruppenname" auf Seite 547
- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

sysmaint_group - SYSMAINT-Gruppenname

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 diesem Parameter wird der Gruppenname definiert, der die Berechtigung SYSMAINT (Systempflege) besitzt. Mit der Berechtigung SYSMAINT können Operationen zur Pflege aller Datenbanken, die zu einem Exemplar gehören, durchgeführt werden, jedoch ohne direkten Zugriff auf Daten.

Achtung: Dieser Parameter muss für Clients unter Windows 98 den Wert NULL besitzen, wenn Systemschutz verwendet wird (d. h., die Authentifizierung ist CLIENT, SERVER, DCS oder eine andere gültige Authentifizierung). Der Grund hierfür ist, dass Windows 98 keine Gruppeninformationen speichert und somit keine Möglichkeit bietet, festzustellen, ob ein Benutzer ein Mitglied einer bestimmten SYSMAINT-Gruppe ist. Bei Angabe eines Gruppennamens kann keiner der Benutzer dieser Gruppe angehören.

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. Sie können auch das Notizbuch **Exemplar konfigurieren** über die DB2-Steuerzentrale aufrufen und den Parameter über das Notizbuch definieren.

Zugehörige Referenzen:

- „sysadm_group - SYSADM-Gruppenname" auf Seite 546
- „sysctrl_group - SYSCTRL-Gruppenname" auf Seite 547
- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

sysmon_group - Berechtigungsgruppenname für Systemmonitor

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 diesem Parameter wird der Gruppenname definiert, der die Berechtigung SYSMON (Systemmonitor) besitzt. Benutzer, die über die Berechtigung SYSMON auf der Exemplarebene verfügen, können über den Datenbanksystemmonitor Momentaufnahmen von einem Datenbankmanagerexemplar oder seinen Datenbank erstellen. Die Berechtigung SYSMON ermöglicht die Ausführung der folgenden Befehle:

- GET DATABASE MANAGER MONITOR SWITCHES
- GET MONITOR SWITCHES
- GET SNAPSHOT
- LIST ACTIVE DATABASES
- LIST APPLICATIONS
- LIST DCS APPLICATIONS
- RESET MONITOR
- UPDATE MONITOR SWITCHES

Benutzer mit der Berechtigung SYSADM, SYSCTRL oder SYSMAINT haben automatisch die Möglichkeit, Momentaufnahmen über den Datenbanksystemmonitor zu erstellen und diese Befehle zu verwenden.

Achtung: Dieser Parameter muss für Clients unter Windows 98 den Wert NULL besitzen, wenn Systemschutz verwendet wird (d. h., die Authentifizierung ist CLIENT, SERVER, DCS oder eine andere gültige Authentifizierung). Der Grund hierfür ist, dass Windows 98 keine Gruppeninformationen speichert und somit keine Möglichkeit bietet, festzustellen, ob ein Benutzer ein Mitglied einer bestimmten SYSMON-Gruppe ist. Bei Angabe eines Gruppennamens kann keiner der Benutzer dieser Gruppe angehören.

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. Sie können auch das Notizbuch **Exemplar konfigurieren** über die DB2-Steuerzentrale aufrufen und den Parameter über das Notizbuch definieren.

Zugehörige Referenzen:

- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

trust_allclnts - Alle Clients akzeptieren

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.

Dieser Parameter und der Parameter *trust_clntauth* werden verwendet, um festzustellen, wo Benutzerberechtigungen für die Datenbankumgebung überprüft werden.

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 Client-Seite ausführen. Alle anderen Clients müssen eine Benutzer-ID und ein Kennwort bereitstellen, deren Gültigkeit vom Server überprüft 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.

Zugehörige Referenzen:

- „authentication - Authentifizierungstyp" auf Seite 538
- „trust_clntauth - Authentifizierung gesicherter Clients" auf Seite 550
- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

trust_clntauth - Authentifizierung gesicherter Clients

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]

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.

Ist dieser Parameter auf CLIENT (die Standardeinstellung) gesetzt, 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.

Der numerische Wert für CLIENT ist 0. Der numerische Wert für SERVER ist 1.

Zugehörige Referenzen:

- „authentication - Authentifizierungstyp" auf Seite 538
- „trust_allclnts - Alle Clients akzeptieren" auf Seite 549
- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

use_sna_auth - SNA-Authentifizierung verwenden

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 *use_sna_auth* auf *yes* und der Parameter *authentication* auf *server* gesetzt ist, werden am Server eingehende Verbindungen, die das SNA-Protokoll mit Sicherheitseinstufung SAME oder PROGRAM verwenden, nur auf der SNA-Schicht authentifiziert und nicht von DB2.

Zugehörige Referenzen:

- „authentication - Authentifizierungstyp" auf Seite 538
- „GET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „RESET DATABASE MANAGER CONFIGURATION Command" in *Command Reference*
- „UPDATE DATABASE MANAGER CONFIGURATION Command" in *Command Reference*

DB2-Verwaltungsserver

Die folgenden Parameter beziehen sich auf den DB2-Verwaltungsserver (DAS):

- „authentication - DAS-Authentifizierungstyp"
- „contact_host - Speicherposition der Liste mit Ansprechpartnern" auf Seite 553
- „das_codepage - DAS-Codepage" auf Seite 553
- „das_territory - DAS-Gebiet" auf Seite 554
- „dasadm_group - DASADM-Gruppenname" auf Seite 554
- „db2system - Name des DB2-Serversystems" auf Seite 555
- „discover - DAS-Discovery-Modus" auf Seite 555
- „exec_exp_task - Verfallene Tasks ausführen" auf Seite 556
- „jdk_64_path - Installationspfad für Software Developer's Kit (64 Bit) für Java auf DAS" auf Seite 557
- „jdk_path - Installationspfad für Software Developer's Kit für Java auf DAS" auf Seite 557
- „sched_enable - Schedulermodus" auf Seite 558
- „sched_userid - Benutzer-ID für Scheduler" auf Seite 558
- „smtp_server - SMTP-Server" auf Seite 559
- „toolscat_db - Toolskatalogdatenbank" auf Seite 559
- „toolscat_inst - Exemplar der Toolskatalogdatenbank" auf Seite 560
- „toolscat_schema - Schema der Toolskatalogdatenbank" auf Seite 561

authentication - DAS-Authentifizierungstyp

Konfigurationstyp	DB2-Verwaltungsserver
Gilt für	DB2-Verwaltungsserver
Parametertyp	Konfigurierbar
Standardwert [Bereich]	SERVER_ENCRYPT [SERVER_ENCRYPT; KERBEROS_ENCRYPT]

Mit diesem Parameter wird festgelegt, wie und wo die Authentifizierung eines Benutzers stattfindet.

Wird als Wert SERVER_ENCRYPT angegeben, werden die Benutzer-ID und das Kennwort vom Client an den Server gesendet, damit die Authentifizierung auf dem Server ausgeführt werden kann. Über das Netzwerk versendete Kennwörter werden verschlüsselt.

Der Wert KERBEROS_ENCRYPT bedeutet, dass die Authentifizierung auf einem Kerberos-Server mit Hilfe des Kerberos-Sicherheitsprotokolls für Authentifizierung ausgeführt wird.

Anmerkung: Der Authentifizierungstyp KERBEROS_ENCRYPT wird nur auf Servern unterstützt, auf denen Windows 2000 ausgeführt wird.

Dieser Parameter kann nur über einen Befehlszeilenprozessor (CLP) der Version 8 aktualisiert werden.

Zugehörige Referenzen:

- „GET ADMIN CONFIGURATION Command" in *Command Reference*

- „RESET ADMIN CONFIGURATION Command" in *Command Reference*
- „UPDATE ADMIN CONFIGURATION Command" in *Command Reference*

contact_host - Speicherposition der Liste mit Ansprechpartnern

Konfigurationstyp	DB2-Verwaltungsserver
Gilt für	DB2-Verwaltungsserver
Parametertyp	Online konfigurierbar
Weitergabeklasse	Sofort
Standardwert [Bereich]	NULL [beliebiger gültiger TCP/IP-Hostname eines DB2-Verwaltungsservers der Version 8]

Dieser Parameter gibt die Speicherposition der Ansprechpartnerinformationen an, die der Scheduler und der Diagnosemonitor für Benachrichtigungen verwenden. Die Position muss der TCP/IP-Hostname eines DB2-Verwaltungsservers sein. Wenn sich *contact_host* auf einem fernen Datenbankverwaltungsserver (DAS) befindet, kann eine Ansprechpartnerliste von mehreren DB2-Verwaltungsservern gemeinsam benutzt werden. Wenn für *contact_host* kein Wert angegeben wird, nimmt der DAS an, dass die Ansprechpartnerinformationen lokal sind. Dieser Parameter kann nur über einen Befehlszeilenprozessor (CLP) der Version 8 aktualisiert werden.

Zugehörige Referenzen:

- „GET ADMIN CONFIGURATION Command" in *Command Reference*
- „RESET ADMIN CONFIGURATION Command" in *Command Reference*
- „UPDATE ADMIN CONFIGURATION Command" in *Command Reference*

das_codepage - DAS-Codepage

Konfigurationstyp	DB2-Verwaltungsserver
Gilt für	DB2-Verwaltungsserver
Parametertyp	Online konfigurierbar
Weitergabeklasse	Sofort
Standardwert [Bereich]	NULL [beliebige gültige DB2-Codepage]

Dieser Parameter zeigt die vom DB2-Verwaltungsserver verwendete Codepage an. Wenn der Parameter auf NULL gesetzt ist, wird die Standardcodepage des Systems verwendet. Dieser Parameter sollte mit der Ländereinstellung der lokalen DB2-Exemplare kompatibel sein. Andernfalls kann der DB2-Verwaltungsserver nicht mit den DB2-Exemplaren kommunizieren.

Dieser Parameter kann nur über einen Befehlszeilenprozessor (CLP) der Version 8 aktualisiert werden.

Zugehörige Referenzen:

- „GET ADMIN CONFIGURATION Command" in *Command Reference*
- „RESET ADMIN CONFIGURATION Command" in *Command Reference*
- „UPDATE ADMIN CONFIGURATION Command" in *Command Reference*
- „das_territory - DAS-Gebiet" auf Seite 554

das_territory - DAS-Gebiet

Konfigurationstyp	DB2-Verwaltungsserver
Gilt für	DB2-Verwaltungsserver
Parametertyp	Online konfigurierbar
Weitergabeklasse	Sofort
Standardwert [Bereich]	NULL [beliebiges gültiges DB2-Gebiet]

Diese Parameter zeigt das vom DB2-Verwaltungsserver verwendete Gebiet an. 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.

Zugehörige Referenzen:

- „GET ADMIN CONFIGURATION Command" in *Command Reference*
- „RESET ADMIN CONFIGURATION Command" in *Command Reference*
- „UPDATE ADMIN CONFIGURATION Command" in *Command Reference*
- „das_codepage - DAS-Codepage" auf Seite 553

dasadm_group - DASADM-Gruppenname

Konfigurationstyp	DB2-Verwaltungsserver
Gilt für	DB2-Verwaltungsserver
Parametertyp	Konfigurierbar
Standardwert [Bereich]	NULL [beliebiger gültiger Gruppenname]

Die Berechtigung DASADM (DAS-Verwaltung) ist die höchste Berechtigungsstufe innerhalb des Datenbankverwaltungsservers (DAS). Dieser Parameter definiert den Gruppennamen mit DASADM-Berechtigung für den DAS.

Die Berechtigung DASADM wird von den in einer bestimmten Betriebsumgebung verwendeten Sicherheitseinrichtungen ermittelt.

- Unter den Betriebssystemen Windows NT und Windows 2000 kann dieser Parameter auf eine beliebige lokale Gruppe gesetzt werden, deren Name aus 8 oder weniger Zeichen besteht und die in der Sicherheitsdatenbank von Windows NT und Windows 2000 definiert ist. Wenn für diesen Parameter der Wert „NULL" angegeben wird, haben alle Mitglieder der Gruppe **Administratoren** die Berechtigung DASADM.
- Wenn auf UNIX-basierten Systemen der Wert „NULL" für diesen Parameter angegeben wird, wird die Primärgruppe des Exemplareigners standardmäßig als DASADM-Gruppe verwendet.

  Ist der Wert nicht „NULL", kann als DASADM-Gruppe jeder gültige UNIX-Gruppenname definiert werden.

Dieser Parameter kann nur über einen Befehlszeilenprozessor (CLP) der Version 8 aktualisiert werden.

Zugehörige Referenzen:

- „GET ADMIN CONFIGURATION Command" in *Command Reference*
- „RESET ADMIN CONFIGURATION Command" in *Command Reference*
- „UPDATE ADMIN CONFIGURATION Command" in *Command Reference*

db2system - Name des DB2-Serversystems

Konfigurationstyp	DB2-Verwaltungsserver
Gilt für	DB2-Verwaltungsserver
Parametertyp	Online konfigurierbar
Standardwert [Bereich]	TCP/IP-Hostname [beliebiger gültiger Systemname]

Dieser Parameter gibt den Namen an, der von Ihren Benutzern und Datenbankadministratoren verwendet wird, um das DB2-Serversystems anzugeben. Dieser Name sollte nach Möglichkeit innerhalb Ihres Netzwerks eindeutig sein.

Dieser Name wird auf der Systemebene der Objektbaumstruktur in der Steuerzentrale angezeigt, um Administratoren bei der Identifizierung von Server-Systemen zu helfen, die von der Steuerzentrale aus verwaltet werden können.

Bei Verwendung der Funktion zum Durchsuchen des Netzwerks des Konfigurationsassistenten gibt DB2-Discovery diesen Namen zurück und er wird auf der Systemebene der sich ergebenden Baumstruktur angezeigt. 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 SETUP-Programm 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.

Zugehörige Referenzen:

- „GET ADMIN CONFIGURATION Command" in *Command Reference*
- „RESET ADMIN CONFIGURATION Command" in *Command Reference*
- „UPDATE ADMIN CONFIGURATION Command" in *Command Reference*
- „discover - DAS-Discovery-Modus" auf Seite 555

discover - DAS-Discovery-Modus

Konfigurationstyp	DB2-Verwaltungsserver
Gilt für	DB2-Verwaltungsserver
Parametertyp	Online konfigurierbar
Weitergabeklasse	Sofort
Standardwert [Bereich]	SEARCH [DISABLE; KNOWN; SEARCH]

Aus der Perspektive eines Verwaltungsservers bestimmt dieser Konfigurationsparameter den Typ des Discovery-Modus, der beim Start des DB2-Verwaltungsservers gestartet wird.

- Wenn für *discover* SEARCH angegeben ist, bearbeitet der Verwaltungsserver Discovery-Anforderungen mit dem Parameterwert SEARCH von Clients. SEARCH stellt eine Implikation der durch Discovery mit dem Parameter KNOWN bereitgestellten Funktionalität zur Verfügung. Wenn für *discover* SEARCH angegeben ist, bearbeitet der Verwaltungsserver sowohl Discovery-Anforderungen mit dem Parameterwert SEARCH als auch Discovery-Anforderungen mit dem Parameterwert KNOWN von Clients.
- Wenn für *discover* KNOWN angegeben ist, bearbeitet der Verwaltungsserver nur Discovery-Anforderungen mit dem Parameterwert KNOWN von Clients.
- Wenn für *discover* DISABLE angegeben ist, bearbeitet der Verwaltungsserver keine Discovery-Anforderungen. Die Informationen für dieses Serversystem sind im Wesentlichen vor Clients verdeckt.

Der Standard-Discovery-Modus ist SEARCH.

Dieser Parameter kann nur über einen Befehlszeilenprozessor (CLP) der Version 8 aktualisiert werden.

Zugehörige Referenzen:
- „GET ADMIN CONFIGURATION Command" in *Command Reference*
- „RESET ADMIN CONFIGURATION Command" in *Command Reference*
- „UPDATE ADMIN CONFIGURATION Command" in *Command Reference*
- „db2system - Name des DB2-Serversystems" auf Seite 555

exec_exp_task - Verfallene Tasks ausführen

Konfigurationstyp	DB2-Verwaltungsserver
Gilt für	DB2-Verwaltungsserver
Parametertyp	Konfigurierbar
Standardwert [Bereich]	No [Yes; No]

Dieser Parameter gibt an, ob der Scheduler Tasks ausführt, die in der Vergangenheit terminiert, bislang jedoch noch nicht ausgeführt wurden. 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.

Dieser Parameter kann nur über einen Befehlszeilenprozessor (CLP) der Version 8 aktualisiert werden.

Zugehörige Referenzen:
- „GET ADMIN CONFIGURATION Command" in *Command Reference*
- „RESET ADMIN CONFIGURATION Command" in *Command Reference*
- „UPDATE ADMIN CONFIGURATION Command" in *Command Reference*
- „sched_enable - Schedulermodus" auf Seite 558
- „toolscat_inst - Exemplar der Toolskatalogdatenbank" auf Seite 560
- „toolscat_db - Toolskatalogdatenbank" auf Seite 559

- „toolscat_schema - Schema der Toolskatalogdatenbank" auf Seite 561
- „smtp_server - SMTP-Server" auf Seite 559
- „sched_userid - Benutzer-ID für Scheduler" auf Seite 558

jdk_64_path - Installationspfad für Software Developer's Kit (64 Bit) für Java auf DAS

Konfigurationstyp	DB2-Verwaltungsserver
Gilt für	DB2-Verwaltungsserver
Parametertyp	Online konfigurierbar
Weitergabeklasse	Sofort
Standardwert [Bereich]	NULL [beliebiger gültiger Pfad]

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-Verwaltungsserver-funktionen auszuführen.

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 errechnet. Dieser Parameter wird nur auf den Plattformen verwendet, die 32- und 64-Bit-Exemplare unterstützen. Diese Plattformen sind auch bekannt als 64-Bit-Hybridplattformen und umfassen AIX, HP-UX und die Solaris-Betriebsumgebung. 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.

Zugehörige Referenzen:
- „GET ADMIN CONFIGURATION Command" in *Command Reference*
- „RESET ADMIN CONFIGURATION Command" in *Command Reference*
- „UPDATE ADMIN CONFIGURATION Command" in *Command Reference*

jdk_path - Installationspfad für Software Developer's Kit für Java auf DAS

Konfigurationstyp	DB2-Verwaltungsserver
Gilt für	DB2-Verwaltungsserver
Parametertyp	Online konfigurierbar
Weitergabeklasse	Sofort
Standardwert [Bereich]	Java-Standardinstallationspfad [beliebiger gültiger Pfad]

Dieser Parameter gibt das Verzeichnis an, in dem das Software Developer's Kit (SDK) für Java installiert ist, das zu verwenden ist, um DB2-Verwaltungsserver-

funktionen auszuführen. Die vom Java-Interpreter verwendeten Umgebungsvariablen werden aus dem Wert dieses Parameters errechnet.

In 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.

Zugehörige Referenzen:

- „GET ADMIN CONFIGURATION Command" in *Command Reference*
- „RESET ADMIN CONFIGURATION Command" in *Command Reference*
- „UPDATE ADMIN CONFIGURATION Command" in *Command Reference*

sched_enable - Schedulermodus

Konfigurationstyp	DB2-Verwaltungsserver
Gilt für	DB2-Verwaltungsserver
Parametertyp	Konfigurierbar
Standardwert [Bereich]	Off [On; Off]

Mit diesem Parameter wird angegeben, ob der Scheduler vom Verwaltungsserver gestartet wird oder nicht. Der Scheduler ermöglicht Tools wie der Taskzentrale das Terminieren und Ausführen von Tasks auf dem Verwaltungsserver.

Dieser Parameter kann nur über einen Befehlszeilenprozessor (CLP) der Version 8 aktualisiert werden.

Zugehörige Referenzen:

- „GET ADMIN CONFIGURATION Command" in *Command Reference*
- „RESET ADMIN CONFIGURATION Command" in *Command Reference*
- „UPDATE ADMIN CONFIGURATION Command" in *Command Reference*
- „toolscat_inst - Exemplar der Toolskatalogdatenbank" auf Seite 560
- „toolscat_db - Toolskatalogdatenbank" auf Seite 559
- „toolscat_schema - Schema der Toolskatalogdatenbank" auf Seite 561
- „smtp_server - SMTP-Server" auf Seite 559
- „exec_exp_task - Verfallene Tasks ausführen" auf Seite 556
- „sched_userid - Benutzer-ID für Scheduler" auf Seite 558

sched_userid - Benutzer-ID für Scheduler

Konfigurationstyp	DB2-Verwaltungsserver
Gilt für	DB2-Verwaltungsserver
Parametertyp	Informativ
Standardwert [Bereich]	NULL [beliebige gültige Benutzer-ID]

Dieser Parameter gibt die Benutzer-ID an, die der Scheduler verwendet, um eine Verbindung zur Toolskatalogdatenbank herzustellen. Der Parameter ist nur dann relevant, wenn die Toolskatalogdatenbank eine ferne Datenbank für den DB2-Verwaltungsserver ist.

Die vom Scheduler für eine Verbindung zur fernen Toolskatalogdatenbank verwendete Benutzer-ID und das Kennwort werden mit dem Befehl **db2admin** angegeben.

Zugehörige Referenzen:

- „GET ADMIN CONFIGURATION Command" in *Command Reference*
- „sched_enable - Schedulermodus" auf Seite 558
- „toolscat_inst - Exemplar der Toolskatalogdatenbank" auf Seite 560
- „toolscat_db - Toolskatalogdatenbank" auf Seite 559
- „toolscat_schema - Schema der Toolskatalogdatenbank" auf Seite 561
- „smtp_server - SMTP-Server" auf Seite 559
- „exec_exp_task - Verfallene Tasks ausführen" auf Seite 556

smtp_server - SMTP-Server

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.

Wenn der Scheduler aktiviert ist, gibt dieser Parameter den SMTP-Server an, den der Scheduler zum Versenden von E-Mail- und Pagerbenachrichtigungen verwendet.

Dieser Parameter kann nur über einen Befehlszeilenprozessor (CLP) der Version 8 aktualisiert werden.

Zugehörige Referenzen:

- „GET ADMIN CONFIGURATION Command" in *Command Reference*
- „RESET ADMIN CONFIGURATION Command" in *Command Reference*
- „UPDATE ADMIN CONFIGURATION Command" in *Command Reference*
- „sched_enable - Schedulermodus" auf Seite 558
- „toolscat_inst - Exemplar der Toolskatalogdatenbank" auf Seite 560
- „toolscat_db - Toolskatalogdatenbank" auf Seite 559
- „toolscat_schema - Schema der Toolskatalogdatenbank" auf Seite 561
- „exec_exp_task - Verfallene Tasks ausführen" auf Seite 556
- „sched_userid - Benutzer-ID für Scheduler" auf Seite 558

toolscat_db - Toolskatalogdatenbank

Konfigurationstyp	DB2-Verwaltungsserver
Gilt für	DB2-Verwaltungsserver

Parametertyp	Konfigurierbar
Standardwert [Bereich]	NULL [beliebiger gültiger Datenbankaliasname]

Dieser Parameter gibt die vom Scheduler verwendete Toolskatalogdatenbank an. Diese Datenbank muss sich im Datenbankverzeichnis des Exemplars befinden, das vom Parameter *toolscat_inst* angegeben ist.

Dieser Parameter kann nur über einen Befehlszeilenprozessor (CLP) der Version 8 aktualisiert werden.

Zugehörige Referenzen:

- „GET ADMIN CONFIGURATION Command" in *Command Reference*
- „RESET ADMIN CONFIGURATION Command" in *Command Reference*
- „UPDATE ADMIN CONFIGURATION Command" in *Command Reference*
- „sched_enable - Schedulermodus" auf Seite 558
- „toolscat_inst - Exemplar der Toolskatalogdatenbank" auf Seite 560
- „toolscat_schema - Schema der Toolskatalogdatenbank" auf Seite 561
- „smtp_server - SMTP-Server" auf Seite 559
- „exec_exp_task - Verfallene Tasks ausführen" auf Seite 556
- „sched_userid - Benutzer-ID für Scheduler" auf Seite 558

toolscat_inst - Exemplar der Toolskatalogdatenbank

Konfigurationstyp	DB2-Verwaltungsserver
Gilt für	DB2-Verwaltungsserver
Parametertyp	Konfigurierbar
Standardwert [Bereich]	NULL [beliebiges gültiges Exemplar]

Dieser Parameter gibt den Exemplarnamen an, der vom Scheduler verwendet wird. Außerdem werden toolscat_db und toolscat_schema angegeben, um die Toolskatalogdatenbank zu kennzeichnen. Die Toolskatalogdatenbank (Tools Catalog) enthält die Taskinformationen, die von der Taskzentrale und der Steuerzentrale erstellt werden. Diese Toolskatalogdatenbank muss sich im Datenbankverzeichnis des Exemplars befinden, das von diesem Konfigurationsparameter angegeben wird. Die Datenbank kann lokal oder fern sein. Bei Toolskatalogdatenbank muss das Exemplar für TCP/IP konfiguriert sein. Bei einer fernen Datenbank muss der Knoten, der im Datenbankverzeichnis katalogisiert wird, ein TCP/IP-Knoten sein.

Dieser Parameter kann nur über einen Befehlszeilenprozessor (CLP) der Version 8 aktualisiert werden.

Zugehörige Tasks:

- „Einrichten und Konfigurieren der Toolskatalogdatenbank und des DAS-Scheduler" in *Systemverwaltung: Implementierung*

Zugehörige Referenzen:

- „GET ADMIN CONFIGURATION Command" in *Command Reference*
- „RESET ADMIN CONFIGURATION Command" in *Command Reference*
- „UPDATE ADMIN CONFIGURATION Command" in *Command Reference*
- „sched_enable - Schedulermodus" auf Seite 558

- „toolscat_db - Toolskatalogdatenbank" auf Seite 559
- „toolscat_schema - Schema der Toolskatalogdatenbank" auf Seite 561
- „smtp_server - SMTP-Server" auf Seite 559
- „exec_exp_task - Verfallene Tasks ausführen" auf Seite 556
- „sched_userid - Benutzer-ID für Scheduler" auf Seite 558

toolscat_schema - Schema der Toolskatalogdatenbank

Konfigurationstyp	DB2-Verwaltungsserver
Gilt für	DB2-Verwaltungsserver
Parametertyp	Konfigurierbar
Standardwert [Bereich]	NULL [beliebiges gültiges Schema]

Dieser Parameter gibt das Schema der vom Scheduler verwendeten Toolskatalogdatenbank an. 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.

Zugehörige Referenzen:

- „GET ADMIN CONFIGURATION Command" in *Command Reference*
- „RESET ADMIN CONFIGURATION Command" in *Command Reference*
- „UPDATE ADMIN CONFIGURATION Command" in *Command Reference*
- „sched_enable - Schedulermodus" auf Seite 558
- „toolscat_inst - Exemplar der Toolskatalogdatenbank" auf Seite 560
- „toolscat_db - Toolskatalogdatenbank" auf Seite 559
- „smtp_server - SMTP-Server" auf Seite 559
- „exec_exp_task - Verfallene Tasks ausführen" auf Seite 556
- „sched_userid - Benutzer-ID für Scheduler" auf Seite 558

Teil 4. Anhänge und Schlussteil

Anhang A. DB2-Registriervariablen und DB2-Umgebungsvariablen

Dieses Kapitel beschreibt die Verwendung der Registrier- und Umgebungsvariablen und listet die Variablen nach einzelnen Kategorien mit einer Erläuterung ihrer Syntax und Verwendung auf.

DB2-Registriervariablen und DB2-Umgebungsvariablen

Dieser Abschnitt listet die Registriervariablen und Umgebungsvariablen für DB2® auf, die Sie vielleicht kennen sollten, um DB2 einsatzfähig und bereit zu machen. Zu jeder Variablen gibt es eine kurze Beschreibung. Einige Variablen sind für Ihre Umgebung möglicherweise nicht relevant.

Zum Anzeigen einer Liste aller unterstützten Registriervariablen führen Sie den folgenden Befehl aus:

```
db2set -lr
```

Zum Ändern des Werts für eine Variable im aktuellen Exemplar bzw. im Standardexemplar führen Sie den folgenden Befehl aus:

```
db2set name_der_registriervariablen=neuer_wert
```

Ob die DB2-Umgebungsvariablen DB2INSTANCE, DB2NODE, DB2PATH und DB2INSTPROF in den DB2-Profilregistrierdatenbanken gespeichert werden, hängt vom Betriebssystem ab. Zur Aktualisierung dieser Umgebungsvariablen verwenden Sie den Befehl `set`. Diese Änderungen werden nach dem nächsten Neustart des Systems wirksam. Auf UNIX®-Plattformen können Sie den Befehl `export` anstelle des Befehls `set` verwenden.

Sie müssen die geänderten Werte für die Registriervariablen definieren, bevor Sie den Befehl DB2START ausführen.

Anmerkung: Wenn eine Registriervariable Boolesche Werte als Argumente erfordert, sind einerseits die Werte YES, 1 und ON sowie andererseits die Werte NO, 0 und OFF äquivalent. Für jede Variable können Sie einen beliebigen der entsprechenden äquivalenten Werte angeben.

Zugehörige Konzepte:

- „Umgebungsvariablen und die Profilregistrierdatenbank" in *Systemverwaltung: Implementierung*

Zugehörige Tasks:

- „Definieren von DB2-Registriervariablen auf Benutzerebene in der LDAP-Umgebung" in *Systemverwaltung: Implementierung*

Zugehörige Referenzen:

- „Allgemeine Registriervariablen" auf Seite 566
- „Systemumgebungsvariablen" auf Seite 569
- „Kommunikationsvariablen" auf Seite 573
- „Befehlszeilenvariablen" auf Seite 577

Registrier- und Umgebungsvariablen nach Kategorie

Die folgenden Abschnitte listen die Registrier- und Umgebungsvariablen nach den verschiedenen Aspekten des Datenbankmanagers bzw. der Datenbankfunktion auf, die sie beeinflussen.

Allgemeine Registriervariablen

Tabelle 46. Allgemeine Registriervariablen

Variablenname	Betriebssystem	Werte
Beschreibung		
DB2ACCOUNT	Alle	Standardwert=Null
Die Abrechnungszeichenfolge für den Benutzer, die an den fernen Host gesendet wird. Einzelheiten siehe *DB2 Connect Benutzerhandbuch*.		
DB2BIDI	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. Im Anhang zur Unterstützung von Landessprachen finden Sie weitere Informationen zur bidirektionalen Sprachenunterstützung.		
DB2CODEPAGE	Alle	Standardwert: abgeleitet aus dem vom Betriebssystem definierten Landescode
Gibt die Codepage der Daten an, die an DB2 für Datenbankclientanwendungen übergeben werden. Der Benutzer sollte DB2CODEPAGE nicht definieren, sofern er nicht in der DB2-Dokumentation oder 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 Daten zur Codepage automatisch aus dem Betriebssystem abruft.		
DB2_COLLECT_TS_REC_INFO	Alle	Standardwert=OFF Werte: YES oder NO
Diese Variable gibt an, ob DB2 alle Protokolldateien bei einer aktualisierenden Wiederherstellung 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, so dass die Informationen, die zum Überspringen von Protokolldateien erforderlich sind, erfasst werden.		
DB2CONSOLECP	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.		
DB2COUNTRY	Windows	Standardwert=null Werte: alle gültigen numerischen Codes für Land, Gebiet oder Region

Tabelle 46. Allgemeine Registriervariablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
Gibt den Code für Land, Gebiet oder Region der Clientanwendung an. Wenn angegeben, überschreibt dieser Wert die Einstellung des Betriebssystems.		
DB2DBDFT	Alle	Standardwert=Null
Gibt den Aliasnamen der Datenbank an, die für implizite Verbindungen zu verwenden ist. Wenn eine Anwendung keine Datenbankverbindung hat, aber SQL-Anweisungen abgesetzt werden, wird eine implizite Verbindung hergestellt, wenn die Umgebungsvariable DB2DBDFT mit einer Standarddatenbank definiert wurde.		
DB2DBMSADDR	32-Bit-Windows-Betriebssysteme	Standardwert= 0x20000000 für Windows NT Wert: 0x20000000 bis 0xB0000000 in Inkrementen von 0x10000
Gibt die Standardadresse des Datenbankmanagers für gemeinsam benutzten Speicher im Hexadezimalformat an. Wenn der Befehl *db2start* aufgrund einer Adresskollision beim gemeinsam benutzen Speicher fehlschlägt, kann diese Registriervariable geändert werden, um das Datenbankmanagerexemplar zu zwingen, den gemeinsam benutzen Speicher an einer anderen Adresse anzulegen.		
DB2DISCOVERYTIME	Windows-Betriebssysteme	Standardwert = 40 Sek. Minimum = 20 Sek.
Gibt die Zeitdauer an, die der Discovery-Prozess SEARCH nach DB2-Systemen sucht.		
DB2_FORCE_APP_ON_MAX_LOG	Alle	Standardwert: TRUE Werte: TRUE, FALSE
Gibt an, was geschieht, wenn der Wert des Konfigurationsparameters MAX_LOG überschritten wird. Wenn TRUE, wird die Anwendung zwangsweise von der Datenbank getrennt und die Arbeitseinheit rückgängig gemacht. Wenn FALSE, schlägt die aktuelle Anweisung fehl. Die Anwendung kann weiterhin von vorherigen Anweisungen in der Arbeitseinheit fertiggestellte Arbeit festschreiben oder sie kann die fertiggestellte Arbeit rückgängig machen, um die Arbeitseinheit zurückzusetzen.		
DB2GRAPHICUNICODESERVER	Alle	Standardwert=OFF Werte: ON oder OFF
Diese Registriervariable 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 Registriervariable 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	Alle	Standardwert = aktuelles Verzeichnis
Gibt einen Pfad an, der bei der Verarbeitung der SQL-Anweisung INCLUDE textdatei während der Ausführung des DB2-Befehls PREP zu verwenden ist. Er gibt eine Liste von Verzeichnissen an, in denen sich die INCLUDE-Datei befinden könnte. Im Handbuch *Application Development Guide* finden Sie eine Beschreibung, wie DB2INCLUDE in den verschiedenen vorkompilierten Sprachen verwendet wird.		
DB2INSTDEF	Windows-Betriebssysteme	Standardwert = DB2
Definiert den Wert, der zu verwenden ist, wenn DB2INSTANCE nicht definiert ist.		
DB2INSTOWNER	Windows NT	Standardwert=Null
Die Registriervariable, die bei der ersten Erstellung des Exemplars in der DB2-Profilregistrierdatenbank erstellt wird. Diese Variable wird auf den Namen der Exemplareignermaschine gesetzt.		

Tabelle 46. Allgemeine Registriervariablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
DB2_LIC_STAT_SIZE	Alle	Standardwert=Null Bereich: 0 bis 32 767
Die Registriervariable 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	Alle	Standardwert: NO Werte: YES oder NO
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 Registriervariable ignoriert.		
DB2NBDISCOVERRCVBUFS	Alle	Standardwert=16 Puffer Minimum=16 Puffer
Diese Variable wird für den NetBIOS-Discovery-Prozess (SEARCH) verwendet. Die Variable gibt die Anzahl der gleichzeitigen Discovery-Antworten an, die von einem Client empfangen werden können. Wenn der Client mehr gleichzeitige Antworten empfängt, als von dieser Variable festgelegt, werden die überzähligen Antworten von der NetBIOS-Schicht gelöscht. Der Standardwert gibt 16 NetBIOS-Empfangspuffer an. Wenn ein kleinerer Wert als der Standardwert gewählt wird, wird der Standardwert verwendet.		
DB2_OBJECT_TABLE_ENTRIES	Alle	Standardwert=0 Werte: 0–50000
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 werden wird, sollten Sie diese Registriervariable 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 eine Onlinesicherung 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.		
DB2OPTIONS	Alle	Standardwert=Null
Legt Optionen für den Befehlszeilenprozessor fest.		
DB2TERRITORY	Alle	Standardwert: abgeleitet aus dem vom Betriebssystem definierten Landescode
Gibt den Regions- oder Gebietscode der Clientanwendung an, was sich auf die Formate für Datum und Uhrzeit auswirkt.		
DB2_VIEW_REOPT_VALUES	Alle	Standardwert=NO Werte: YES, NO
Diese Variable gibt allen Benutzern die Möglichkeit, die im Cache abgelegten Werte einer reoptimierten SQL-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.		

Zugehörige Konzepte:

• „DB2-Registriervariablen und DB2-Umgebungsvariablen" auf Seite 565

Systemumgebungsvariablen

Tabelle 47. Systemumgebungsvariablen

Variablenname	Betriebssystem	Werte
Beschreibung		
DB2CONNECT_IN_APP_PROCESS	Alle	Standardwert=YES Werte: YES oder NO
Wenn Sie diese Variable auf den Wert NO setzen, werden lokale DB2 Connect-Clients auf einer Maschine mit DB2 Connect Enterprise Edition 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.		
DB2DOMAINLIST	Nur Windows NT-Server	Standardwert=null Werte: Eine Liste mit Windows NT-Domänennamen, die durch Kommata („,”) getrennt werden.
Definiert eine oder mehrere Windows NT-Domänen. Diese Liste definiert die Domänen, denen gegenüber 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 Konfiguration des Datenbankmanagers die Clientauthentifizierung definiert ist, und wird benötigt, wenn eine einmalige Anmeldung (single sign-on) über einen Window NT-Desktop in einer Windows NT-Domänenumgebung erforderlich ist. Diese Registriervariable sollte nur in einer reinen Windows NT-Domänenumgebung mit DB2-Servern und -Clients mit Version 7.1 (oder höher) verwendet werden.		
DB2ENVLIST	UNIX	Standardwert: null
Gibt eine Liste spezifischer Variablennamen entweder für gespeicherte Prozeduren oder für benutzerdefinierte Funktionen an. Standardmäßig filtert der Befehl **db2start** alle Benutzerumgebungsvariablen mit Ausnahme derjenigen heraus, die das Präfix **DB2** oder **db2** haben. Wenn spezifische 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	Alle	Standardwert=DB2INSTDEF unter Windows-Betriebssystemen mit 32-Bit-Architektur
Die Umgebungsvariable, die zur Definition des Exemplars dient, das standardmäßig aktiv ist. Unter UNIX müssen Benutzer einen Wert für DB2INSTANCE festlegen.		
DB2INSTPROF	Windows-Betriebssysteme	Standardwert: null
Die Umgebungsvariable, die zur Angabe der Position des Exemplarverzeichnisses unter Windows-Betriebssystemen dient, wenn sie von DB2PATH abweicht.		
DB2LIBPATH	UNIX	Standardwert: null
DB2 bildet einen eigenen gemeinsamen Bibliothekspfad. Wenn Sie einen Pfad (PATH) in den Bibliothekspfad der Steuerkomponente einfügen wollen (z. B. erfordert eine benutzerdefinierte Funktion unter AIX einen besonderen Eintrag in LIBPATH), müssen Sie die Variable DB2LIBPATH definieren. Der tatsächliche Wert der Variablen DB2LIBPATH wird an das Ende des von DB2 gebildeten gemeinsamen Bibliothekspfad angehängt.		
DB2NODE	Alle	Standardwert: null Werte: 1 bis 999
Dient zur Angabe des logischen Zielknotens eines Datenbankpartitionsservers unter DB2 Enterprise Server Edition, zu dem eine CONNECT- oder ATTACH-Verbindung 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.		

Tabelle 47. Systemumgebungsvariablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
DB2_PARALLEL_IO	Alle	Standardwert: null Werte: • *[:*plattenanzahl*] - bedeutet, dass die parallele E/A für jeden Tabellenbereich aktiviert wird. Nach dem Doppelpunkt kann ein Wert oder ein Symbol angegeben werden, um die Standardanzahl von Platten pro Behälter für alle Tabellenbereiche zu definieren. Der Standardwert 6 (der Wert für eine RAID-5-Einheit) wird verwendet, wenn die Plattenanzahl nicht angegeben wird. • TabellenbereichsID[:*plattenanzahl*],... - eine durch Kommata getrennte Liste definierter Tabellenbereiche. Zur Definition einer Plattenanzahl pro Behälter für diesen Tabellenbereich geben Sie einen Doppelpunkt und einen Wert nach jeder Tabellenbereichs-ID an. Dieser Wert kann ein numerischer Wert oder eines der in der nachfolgenden Tabelle beschriebenen Symbole sein. • Eine Kombination der obigen zwei Wertetypen, wobei jeder Wert durch ein Komma zu trennen ist.

Tabelle 47. Systemumgebungsvariablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
Diese Registriervariable 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 Behälter 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 eines EXTENTSIZE großen Speicherbereichs von Seiten. Wenn diese Registriervariable nicht definiert ist, entspricht der Grad der Parallelität eines Tabellenbereichs der Anzahl der Behälter des Tabellenbereichs. Wenn DB2_PARALLEL_IO zum Beispiel auf den Wert null gesetzt wird und ein Tabellenbereich vier Behälter besitzt, werden vier Vorablesezugriffsanforderungen in der Größe von EXTENTSIZE abgesetzt. Wenn diese Registriervariable definiert ist, entspricht der Grad der Parallelität des Tabellenbereichs dem Verhältnis zwischen dem Wert für PREFETCHSIZE und dem Wert für EXTENTSIZE dieses Tabellenbereichs. Wenn DB2_PARALLEL_IO zum Beispiel für einen Tabellenbereich definiert ist, dessen PREFETCHSIZE-Wert 160 und dessen EXTENTSIZE-Wert 32 Seiten betragen, werden fünf EXTENTSIZE große Vorablesezugriffsanforderungen abgesetzt. Ein Platzhalterzeichen "*" kann verwendet werden, um DB2 anzuweisen, die E/A-Parallelität für alle Tabellenbereiche auf diese Weise zu berechnen. In E/A-Subsystemen, die eine einheitenübergreifende Verwendung physischer Spindeln (Striping) unter jedem DB2-Behälter unterstützen (z. B. mit einer RAID-Einheit), sollte die Anzahl von Platten unter jedem DB2-Behälter bei der Auswahl eines Werts für PREFETCHSIZE für den Tabellenbereich mit berücksichtigt werden. Der Wert für PREFETCHSIZE sollte nach folgender Gleichung berechnet werden: `PREFETCHSIZE = (Anzahl Behälter) * (Anzahl Platten pro Behälter) * EXTENTSIZE` DB2 berechnet in der Regel den Wert für PREFETCHSIZE eines Tabellenbereichs anhand der obigen Gleichung, wenn der Wert für PREFETCHSIZE des Tabellenbereichs auf AUTOMATIC gesetzt ist. Die Registriervariable DB2_PARALLEL_IO kann verwendet werden, um DB2 die Anzahl von Platten pro Behälter anzugeben. Wenn zum Beispiel DB2_PARALLEL_IO="1:4" definiert wird und Tabellenbereich 1 drei Behälter, den EXTENTSIZE-Wert 32 und den PREFETCHSIZE-Wert AUTOMATIC hat, wird der Wert für PREFETCHSIZE mit 3 * 4 * 32 = 384 berechnet. Die E/A-Parallelität dieses Tabellenbereichs ist 384 / 32 = 12. Wenn der PREFETCHSIZE-Wert eines Tabellenbereichs nicht auf AUTOMATIC gesetzt ist, wird die Information über die Anzahl von Platten pro Behälter nicht verwendet. Von jedem Tabellenbereich, der unter DB2_PARALLEL_IO definiert ist, wird standardmäßig angenommen, dass er sechs Platten pro Behälter besitzt, sofern dies in der Registriervariablen nicht anders angegeben wird. Wenn zum Beispiel DB2_PARALLEL_IO=*,1:3 definiert wird, verwenden alle Tabellenbereiche den Wert 6 als Anzahl der Platten pro Behälter, mit Ausnahme von Tabellenbereich 1, der den Wert 3 verwendet.		
DB2PATH	Windows-Betriebssysteme	Standardwert: (je nach Betriebssystem unterschiedlich)
Die Umgebungsvariable, die zur Angabe des Verzeichnisses dient, in dem das Produkt unter 32-Bit-Windows-Betriebssystemen installiert ist.		
DB2PROCESSORS	Windows-Betriebssysteme	Standardwert: null Werte: 0–n-1 (hierbei ist n = Anzahl der Prozessoren)
Definiert die Prozessaffinitätsmaske für einen bestimmten db2syscs-Prozess. 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, gibt DB2 die API 'SetProcessAffinityMask()' aus. Wird diese Variable nicht angegeben, wird der Prozess 'db2syscs' allen Prozessoren auf der Maschine zugeordnet.		

Tabelle 47. Systemumgebungsvariablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
DB2_USE_PAGE_CONTAINER_TAG	Alle	Standardwert: null Werte: ON, null
Standardmäßig speichert DB2 eine Behälterkennung (engl. container tag) im ersten EXTENTSIZE-Speicherbereich jedes DMS-Behälters, unabhängig davon, ob es sich um eine Datei oder eine Einheit handelt. Die Behälterkennung bildet die Metadaten für den Behälter. Vor DB2 Version 8.1 wurde die Behälterkennung in einer einzigen Seite gespeichert und erforderte so weniger Speicherplatz im Behälter. Wenn die Behälterkennung 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 Registriervariable auf ON setzen, wenn Sie RAID-Einheiten als Behälter 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 Registriervariable zu aktivieren. Zur Aktivierung von Änderungen an dieser Registriervariablen führen Sie den Befehl DB2STOP und anschließend den Befehl DB2START aus.		
DB2_CLPHISTSIZE	Alle	Standardwert: 20 **Anmerkung:** Diese Registrierdatenbankvariable wird während der Installation nicht auf den Standardwert festgelegt. Statt dessen verwendet der Code, der diese Variable verwendet, einen Standardwert von 20, wenn die Registrierdatenbankvariable nicht festgelegt ist oder wenn sie auf einen Wert festgelegt wurde, der sich außerhalb des gültigen Bereichs befindet.Werte: 1–500 einschließlich
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 ab von der Anzahl und Länge der Befehle, die in einer Sitzung ausgeführt werden.		
DB2_CLP_EDITOR	Alle	Standardwert: Windows-Plattformen: 'Editor' UNIX: 'vi' **Anmerkung:** Diese Registrierdatenbankvariable wird während der Installation nicht auf den Standardwert festgelegt. Statt dessen verwendet der Code, der diese Variable verwendet, einen Standardwert, wenn die Registrierdatenbankvariable nicht festgelegt ist.Werte: Alle gültigen Editoren, die sich im Betriebssystempfad befinden.
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.		

Zugehörige Konzepte:

- „DB2-Registriervariablen und DB2-Umgebungsvariablen" auf Seite 565

Kommunikationsvariablen

Tabelle 48. Kommunikationsvariablen

Variablenname	Betriebssystem	Werte
Beschreibung		
DB2CHECKCLIENTINTERVAL	Alle, nur Server	Standardwert=50 Werte: ein numerischer Wert größer als null
Gibt die Häufigkeit von APPC- und TCP/IP-Clientverbindungsprüfungen an. Ermöglicht das frühe Erkennen einer Clientbeendigung, anstatt zu warten, bis die Abfrage abgeschlossen ist. Wird diese Variable auf 0 gesetzt, wird keine Prüfung durchgeführt. Niedrigere Werte bedeuten häufigere Prüfungen. Verwenden Sie 100 als Richtwert für geringe Häufigkeit, 50 für mittlere Häufigkeit und 10 für hohe Häufigkeit. Je häufiger Sie den Clientstatus während der Ausführung einer Datenbankanforderung prüfen, desto länger dauert die Beendigung von Abfragen. Wenn die DB2-Auslastung hoch ist (d. h., wenn viele interne Anforderungen anliegen), hat das Einstellen von DB2CHECKCLIENTINTERVAL auf einen niedrigen Wert eine größere Auswirkung auf die Leistung als in einer Situation, in der die Auslastung niedrig ist und DB2 die meiste Zeit wartet. In DB2 UDB Version 8.1.4 hat DB2CHECKCLIENTINTERVAL den Standardwert 50. Vor Version 8.1.4 war der Standardwert 0.		
DB2COMM	Alle, nur Server	Standardwert=null Werte: eine beliebige Kombination aus APPC, IPXSPX, NETBIOS, NPIPE, TCPIP
Definiert die Kommunikationsmanager, die gestartet werden, wenn der Datenbankmanager gestartet wird. Wenn dieser Parameter nicht definiert wird, werden auf dem Server keine DB2-Kommunikationsmanager gestartet.		
DB2_FORCE_NLS_CACHE	AIX, HP_UX, Solaris Operating Environment	Standardwert=FALSE Werte: TRUE oder FALSE
Dient zur Verhinderung der Möglichkeit von Sperrkonflikten in Multithread-Anwendungen. Wenn diese Registriervariable „TRUE" ist, 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 ausgeschaltet, was in bestimmten Situationen zu einem Leistungsvorteil führt. Diese Einstellung sollte nicht verwendet werden, wenn die Anwendung die länderspezifischen Angaben (Locale) zwischen Verbindungen ändert. In solch einer Situation ist sie wahrscheinlich kaum erforderlich, da Multithread-Anwendungen ihre länderspezifischen Angaben in der Regel nicht ändern, weil dies die Threadstabilität gefährden könnte.		
DB2JD_PORT_NUMBER	Alle	Standardwert=6789 Werte: 1 - 65535
Diese Registriervariable kann zum Überschreiben der Standardportnummer von db2jd verwendet werden. Wenn diese Registriervariable definiert ist, versucht db2jd, diesen Wert als Empfangsport von db2jd zu verwenden, wenn db2jstrt ohne Parameter ausgeführt wird. Wenn diese Registriervariable nicht definiert ist, und kein Portparameter angegeben wird, startet db2jd am Standardport 6789.		
DB2NBADAPTERS	Windows	Standardwert=0 Bereich: 0-15, Mehrere Werte müssen durch Kommas getrennt werden.
Dient zur Definition, welche lokalen Adapter für die DB2-NetBIOS-LAN-Kommunikation zu verwenden sind. Jeder lokale Adapter wird mit Hilfe seiner logischen Adapternummer angegeben.		

Tabelle 48. Kommunikationsvariablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
DB2NBCHECKUPTIME	Nur Windows-Server	Standardwert=1 Minute Werte: 1-720
Definiert das Zeitintervall zwischen den Aufrufen der Prüfprozedur des NetBIOS-Protokolls. Die Prüfzeit wird in Minuten angegeben. Niedrigere Werte stellen sicher, dass die Prüfprozedur des NetBIOS-Protokolls häufiger aktiv wird und Speicher und andere Systemressourcen freigibt, die zurückbleiben, wenn Agenten oder Sitzungen unerwartet beendet werden.		
DB2NBINTRLISTENS	Nur Windows-Server	Standardwert=1 Werte: 1-10 Mehrere Werte müssen durch Kommata getrennt werden.
Definiert die Anzahl der NetBIOS-Befehle listen send (NCBs - NetBIOS-Steuerblöcke), die asynchron in Empfangsbereitschaft für Unterbrechungen ferner Clients abgesetzt werden. Diese Flexibilität ist für "unterbrechungsaktive" Umgebungen gedacht, um sicherzustellen, dass Unterbrechungsaufrufe von fernen Clients eine Verbindung herstellen können, wenn die Server mit der Bedienung anderer ferner Unterbrechungen beschäftigt sind. Durch Definieren eines niedrigeren Werts für DB2NBINTRLISTENS werden NetBIOS-Sitzungen und NetBIOS-Steuerblöcke (NCBs) auf dem Server geschont. In einer Umgebung jedoch, in der Clientunterbrechungen regelmäßig vorkommen, ist ein höherer Wert für DB2NBINTRLISTENS eventuell sinnvoll, um auf unterbrechende Clients flexibler reagieren zu können. **Anmerkung:** Die angegebenen Werte sind positionsabhängig. Sie beziehen sich jeweils auf die entsprechenden Wertpositionen für DB2NBADAPTERS.		
DB2NBRECVBUFFSIZE	Nur Windows-Server	Standardwert=4096 Byte Bereich: 4096-65536
Definiert die Größe der Empfangspuffer des DB2-NetBIOS-Protokolls. Diese Puffer werden den NetBIOS-Empfangssteuerblöcken zugeordnet (NCBs). Niedrigere Werte sparen Serverspeicher, während höhere Werte erforderlich sein können, wenn die Clientdatenübertragungen umfangreicher sind.		
DB2NBBRECVNCBS	Nur Windows-Server	Standardwert=10 Bereich: 1-99
Definiert die Anzahl der NetBIOS-Befehle "receive_any" (NCBs), die der Server beim Betrieb absetzt und verwaltet. Dieser Wert wird je nach der Anzahl ferner Clients, mit denen der Server verbunden ist, angepasst. Niedrigere Werte sparen Serverressourcen. **Anmerkung:** Für jeden benutzten Adapter kann ein eigener eindeutiger Empfangs-NCB-Wert im Parameter DB2NBBRECVNCBS angegeben werden. Die angegebenen Werte sind positionsabhängig. Sie beziehen sich jeweils auf die entsprechenden Wertpositionen für DB2NBADAPTERS.		
DB2NBRESOURCES	Nur Windows-Server	Standardwert=Null
Definiert die Anzahl der NetBIOS-Ressourcen, die zur Verwendung durch DB2 in einer Mehrkontextumgebung zuzuordnen sind. Diese Variable ist auf den Clientbetrieb in einer Mehrkontextumgebung beschränkt.		
DB2NBSENDNCBS	Nur Windows-Server	Standardwert=6 Bereich: 1-720
Definiert die Anzahl der NetBIOS-Befehle send (NCBs), die der Server zur Verwendung empfängt. Dieser Wert kann angepasst werden je nach der Anzahl ferner Clients, mit denen der Server verbunden ist. Durch Definieren eines niedrigeren Werts für DB2NBSENDNCBS werden Serverressourcen geschont. Jedoch kann ein höherer Wert erforderlich werden, um zu vermeiden, dass der Server mit einem send-Befehl an einen fernen Client warten muss, wenn alle anderen send-Befehle in Gebrauch sind.		

Tabelle 48. Kommunikationsvariablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
DB2NBSESSIONS	Nur Windows-Server	Standardwert=Null Bereich: 5-254
Definiert die Anzahl von Sitzungen, die DB2 zur Reservierung für DB2 anfordern soll. Der Wert von DB2NBSESSIONS kann definiert werden, um eine spezifische Sitzung für jeden in DB2NBADAPTERS angegebenen Adapter anzufordern. **Anmerkung:** Die angegebenen Werte sind positionsabhängig. Sie beziehen sich jeweils auf die entsprechenden Wertpositionen für DB2NBADAPTERS.		
DB2NBXTRANCBS	Nur Windows-Server	Standardwert=5 pro Adapter Bereich: 5-254
Definiert die Anzahl von NetBIOS-Befehlen "extra" (NCBs), die der Server reservieren muss, wenn der Befehl **db2start** ausgeführt wird. Der Wert von DB2NBXTRANCBS kann definiert werden, um eine spezifische Sitzung für jeden in DB2NBADAPTERS angegebenen Adapter anzufordern.		
DB2RETRY	Windows	Standardwert=0 Bereich: 0-20 000
Die Anzahl der DB2-Versuche, die APPC-Empfangsfunktion erneut zu starten. Wenn das SNA-Subsystem auf dem Server/Gateway abgestürzt ist, kann die APPC-Empfangsfunktion mit Hilfe dieser Profilvariablen zusammen mit DB2RETRYTIME automatisch erneut gestartet werden, ohne die Clientdatenfernverarbeitung unter Verwendung anderer Protokolle zu unterbrechen. Bei einem derartigen Szenario braucht DB2 nicht gestoppt und erneut gestartet zu werden, um die APPC-Client-Datenfernverarbeitung wiederherzustellen.		
DB2RETRYTIME	Windows	Standardwert=1 Minute Bereich: 0-7 200 Minuten
In Inkrementen von einer Minute die Anzahl Minuten, die DB2 zwischen dem Ausführen aufeinander folgender Wiederholungen zum Starten der APPC-Empfangsfunktion zulässt. Wenn das SNA-Subsystem auf dem Server/Gateway abgestürzt ist, kann die APPC-Empfangsfunktion mit Hilfe dieser Profilvariablen zusammen mit DB2RETRY automatisch erneut gestartet werden, ohne die Client-Datenfernverarbeitung unter Verwendung anderer Protokolle zu unterbrechen. Bei einem derartigen Szenario braucht DB2 nicht gestoppt und erneut gestartet zu werden, um die APPC-Client-Datenfernverarbeitung wiederherzustellen.		
DB2SERVICETPINSTANCE	Windows, AIX und Solaris Operating Environment	Standardwert=null
Dient zur Lösung des durch folgende Ursachen ausgelösten Problems: • Auf einer Maschine sind mehrere Exemplare aktiv. • Ein Exemplar der Version 6 oder der Version 7 auf derselben Maschine versucht, dieselben TP-Namen zu registrieren. Wenn der Befehl **db2start** aufgerufen wird, startet das angegebene Exemplar die APPC-Empfangsfunktionen (Listeners) für die folgenden TP-Namen: • DB2DRDA • x'07'6DB		
DB2SORCVBUF	Alle	Standardwert=65536
Definiert den Wert von TCP/IP-Empfangspuffern unter Windows-Betriebssystemen.		
DB2SOSNDBUF	Alle	Standardwert=65536
Definiert den Wert von TCP/IP-Sendepuffern auf Windows NT-Betriebssystemen.		
DB2SYSPLEX_SERVER	Windows NT und UNIX	Standardwert=null

Tabelle 48. Kommunikationsvariablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
Gibt an, ob beim Herstellen einer Verbindung zu DB2 für OS/390 oder z/OS die SYSPLEX-Ausnutzung aktiviert wird. Ist diese Registriervariable nicht vorhanden (Standardeinstellung), oder wird sie auf einen anderen Wert als Null eingestellt, wird die Ausnutzung aktiviert. Wird diese Registriervariable auf Null (0) eingestellt, wird die Ausnutzung inaktiviert. Bei dieser Einstellung wird die SYSPLEX-Ausnutzung für den Gateway inaktiviert, unabhängig davon, wie der DCS-Datenbankkatalogeintrag angegeben wurde. Weitere Informationen finden Sie in der Beschreibung des Befehls **CATALOG DCS DATABASE** für den Befehlszeilenprozessor.		
DB2TCPCONNMGRS	Alle	Standardwert=1 auf seriellen Maschinen; auf symmetrischen Mehrprozessormaschinen die aufgerundete Quadratwurzel aus der Anzahl Prozessoren bis zu maximal acht Verbindungsmanagern. Werte: 1 bis 8
Wenn die Registriervariable nicht definiert ist, wird die Standardzahl von Verbindungsmanagern erstellt. Wenn die Registriervariable definiert ist, setzt der zugeordnete Wert den Standardwert außer Kraft. Die Zahl der angegebenen TCP/IP-Verbindungsmanager wird bis zu einem Maximum von 8 erstellt. Wenn weniger als einer angegeben wird, wird DB2TCPCONNMGRS auf den Wert 1 gesetzt und eine Warnung protokolliert, dass der Wert nicht innerhalb des gültigen Bereichs liegt. Wenn ein Wert größer als acht angegeben wird, wird DB2TCPCONNMGRS auf den Wert 8 gesetzt und eine Warnung protokolliert, dass der Wert nicht innerhalb des gültigen Bereichs liegt. Werte zwischen 1 und 8 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) auftreten. 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 Verbindungsauf- und -abbau zu einem Leistungsabfall.		
DB2_VI_ENABLE	Windows NT	Standardwert=OFF Werte: ON oder OFF
Gibt an, ob das Kommunikationsprotokoll VI Architecture (VI - Virtual Interface) verwendet werden soll oder nicht. Ist diese Registriervariable `ON`, dann verwendet FCM das VI-Protokoll für die Kommunikation zwischen Knoten. Ist diese Registriervariable `OFF`, dann verwendet FCM das Protokoll TCP/IP für die Kommunikation zwischen Knoten. **Anmerkung:** Der Wert dieser Registriervariablen muss in allen Datenbankpartitionen des Exemplars gleich sein.		
DB2_VI_VIPL	Windows NT	Standardwert = `vipl.dll`
Gibt den Namen der VIPL (Virtual Interface Provider Library) an, die von DB2 verwendet wird. Der in dieser Registriervariablen verwendete Bibliotheksname muss in der Benutzerumgebungsvariablen PATH angegeben werden, um die Bibliothek erfolgreich laden zu können. Die zurzeit unterstützten Implementierungen verwenden alle denselben Bibliotheksnamen.		
DB2_VI_DEVICE	Windows NT	Standardwert=Null Werte: `nic0` oder `VINIC`
Gibt den symbolischen Namen der Einheit oder des VIP-Exemplars (VIP - Virtual Interface Provider) an, die bzw. das der NIC (Network Interface Card) zugeordnet ist. Unabhängige Hardwarelieferanten produzieren jeweils eigene NICs. Pro Windows NT-Maschine ist nur jeweils eine NIC zulässig. Mehrere logische Knoten auf derselben physischen Maschine benutzen die NIC gemeinsam. Der symbolische Name der Einheit „VINIC" muss in Großbuchstaben eingegeben werden und kann nur mit Synfinity Interconnect verwendet werden. Alle sonstigen derzeit unterstützen Implementierungen verwenden „nic0" als symbolischen Namen der Einheit.		

Zugehörige Konzepte:

• „DB2-Registriervariablen und DB2-Umgebungsvariablen" auf Seite 565

Befehlszeilenvariablen

Tabelle 49. Befehlszeilenvariablen

Variablenname	Betriebssystem	Werte
Beschreibung		
DB2BQTIME	Alle	Standardwert=1 Sekunde Maximalwert: 1 Sekunde
Definiert die Zeitdauer, 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	Alle	Standardwert=60 Wiederholungen Minimalwert: 0 Wiederholungen
Definiert die Anzahl der Wiederholungen, mit denen das Front-End des Befehlszeilenprozessors festzustellen versucht, ob der Back-End-Prozess bereits aktiv ist. Dieser Parameter arbeitet in Verbindung mit dem Parameter DB2BQTIME.		
DB2_CLPPROMPT	Alle	Standardwert = Keiner Wird diese Variable nicht definiert, wird „db2 => ” als standardmäßige interaktive CLP-Eingabeaufforderung verwendet. Gültige Werte: Alle Textzeichenfolgen mit einer Länge von weniger als 100 Zeichen, die keinen oder mehrere der folgenden Tokens enthalten: %i, %d, %ia, %da oder %n. Diese Variable braucht nur dann definiert zu werden, wenn die standardmäßige interaktive CLP-Eingabeaufforderung (db2 =>) explizit geändert werden soll.
Mit dieser Registrierdatenbankvariable können Benutzer die Eingabeaufforderung definieren, die im interaktiven Modus des Befehlszeilenprozessors (CLP) verwendet werden soll. Gültige Werte für die Variable: Alle Textzeichenfolgen mit einer Länge von weniger als 100 Zeichen, die keinen oder mehrere der folgenden optionalen Tokens enthalten: %i, %d, %ia, %da oder %n. Wird der interaktive CLP-Modus ausgeführt, wird die zu verwendende Eingabeaufforderung erstellt, indem die in der Registrierdatenbankvariablen DB2_CLPPROMPT angegebene Textzeichenfolge verwendet wird und alle Vorkommnisse der Tokens %i, %d, %ia, %da oder %n durch den lokalen Aliasnamen des jeweils zugeordneten Exemplars, den lokalen Aliasnamen der aktuellen Datenbankverbindung, die Berechtigungs-ID des jeweils zugeordneten Exemplars, die Berechtigungs-ID der aktuellen Datenbankverbindung bzw. eine neue Zeile (d. h. eine Zeilenschaltung) ersetzt wird. **Anmerkungen:** 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. Ist kein Exemplar zugeordnet, wird %ia durch eine leere Zeichenfolge und %i durch den Wert der Registrierdatenbankvariablen DB2INSTANCE ersetzt. Nur auf Windows-Plattformen: Wird die Variable DB2INSTANCE nicht definiert, wird %i durch den Wert der Registrierdatenbankvariablen DB2INSTDEF ersetzt. Wird keine dieser Variablen definiert, wird %i durch eine leere Zeichenfolge ersetzt. 3. Ist keine Datenbankverbindung vorhanden, wird %da durch eine leere Zeichenfolge und %d durch den Wert der Registrierdatenbankvariablen DB2DBDFT ersetzt. Wird 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 Exemplarnamen stets in Großbuchstaben an.		
DB2IQTIME	Alle	Standardwert=5 Sekunden Minimalwert: 1 Sekunde

Tabelle 49. Befehlszeilenvariablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
Definiert die Zeitdauer, die der Back-End-Prozess des Befehlszeilenprozessors an der Eingabewarteschlange darauf wartet, dass der Front-End-Prozess Befehle übergibt.		
DB2RQTIME	Alle	Standardwert=5 Sekunden Minimalwert: 1 Sekunde
Definiert die Zeitdauer, die der Back-End-Prozess des Befehlszeilenprozessors auf eine Anforderung vom Front-End-Prozess wartet.		

Zugehörige Konzepte:

- „DB2-Registriervariablen und DB2-Umgebungsvariablen" auf Seite 565

MPP-Konfigurationsvariablen

Tabelle 50. MPP-Konfigurationsvariablen

Variablenname	Betriebssystem	Werte
DB2ATLD_PWFILE	DB2 UDB ESE unter AIX, Solaris Operating Environment und Windows NT	Standardwert=null Wert: ein Ausdruck für einen Dateipfad
Gibt einen Pfad zu einer Datei an, die ein Kennwort enthält, das bei der Autoloader-Identifikationsüberprüfung verwendet wird. Wenn kein Wert definiert ist, extrahiert Autoloader das Kennwort entweder aus der zugehörigen Konfigurationsdatei oder fordert Sie auf, es anzugeben. Die Verwendung dieser Variablen berücksichtigt Sicherheitsbelange und ermöglicht die Trennung von Autoloader-Konfigurationsdaten von Identifikationsüberprüfungsdaten. Diese Registriervariable ist nicht länger erforderlich, wird jedoch aus Gründen der Abwärtskompatibilität beibehalten.		
DB2CHGPWD_EEE	DB2 UDB ESE unter AIX und Windows NT	Standardwert=null Werte: YES oder NO
Mit dieser Variablen können Sie zulassen, dass andere Benutzer Kennwörter auf ESE-Systemen unter AIX oder Windows NT ändern. Es muss sichergestellt werden, dass die Kennwörter für alle Partitionen oder Knoten mit Hilfe eines Windows NT-Domänencontrollers unter Windows NT oder NIS unter AIX zentral verwaltet werden. Wenn Kennwörter nicht zentral verwaltet werden, bleiben sie möglicherweise nicht für alle Partitionen 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. Um diese globale Registriervariable zu ändern, müssen Sie sich im Stammverzeichnis und im DAS-Exemplar befinden. Diese Variable ist nur erforderlich, wenn Sie das alte Dienstprogramm *db2atld* anstelle des neuen Dienstprogramms LOAD verwenden.		
DB2_FORCE_FCM_BP	AIX	Standardwert=No Werte: Yes oder No

Tabelle 50. MPP-Konfigurationsvariablen (Forts.)

Variablenname	Betriebssystem	Werte
Diese Registriervariable ist für DB2 UDB ESE für AIX mit mehreren logischen Partitionen gültig. Wenn der Befehl DB2START abgesetzt wird, ordnet DB2 die FCM-Puffer entweder aus dem globalen Speicher oder aus einem separaten gemeinsamen Speichersegment zu, wenn nicht genügend globaler Speicher verfügbar ist. Diese Puffer werden von allen FCM-Dämonen für dieses Exemplar auf der gleichen physischen Maschine verwendet. Die Art des zugeordneten Speichers hängt weit gehend von der Anzahl der zu erstellenden FCM-Puffer ab, die durch den Konfigurationsparameter *fcm_num_buffers* des Datenbankmanagers angegeben wird. Wenn die Variable DB2_FORCE_FCM_BP auf den Wert YES gesetzt wird, werden die FCM-Puffer immer in einem separaten Speichersegment erstellt, so dass die Kommunikation zwischen FCM-Dämonen verschiedener logischer Partitionen auf dem gleichen physischen Knoten über den gemeinsamen Speicher erfolgt. Ansonsten kommunizieren FCM-Domänen auf dem gleichen Knoten über UNIX-Sockets. Die Kommunikation über den gemeinsamen Speicher ist schneller, jedoch ist ein gemeinsames Speichersegment weniger für andere Zwecke, insbesondere für Datenbankpufferpools, verfügbar. Die Aktivierung der Registriervariablen DB2_FORCE_FCM_BP verringert also die maximale Größe der Datenbankpufferpools.		
DB2_NUM_FAILOVER_NODES	Alle	Standardwert: 2 Werte: 0 bis zur Anzahl der logischen Knoten
Gibt die Anzahl der Knoten an, die als als Funktionsübernahmeknoten in einer Umgebung mit hoher Verfügbarkeit verwendet werden können. Bei hoher Verfügbarkeit kann ein Knoten beim Auftreten eines Fehlers als zweiter logischer Knoten auf einem anderen Host erneut gestartet werden. Die für diese Variable verwendete Zahl legt fest, wie viel Speicher für FCM-Ressourcen für Funktionsübernahmeknoten reserviert ist. Beispielsweise verfügt Host A über zwei logische Knoten: 1 und 2, und Host B verfügt über zwei logische Knoten: 3 und 4. Nehmen Sie an, dass DB2_NUM_FAILOVER_NODES auf den Wert 2 gesetzt ist. Während der Ausführung von DB2START reservieren Host A und auch Host B genügend Speicher für den FCM, um bis zu vier logische Knoten verwalten zu können. Wenn dann bei einem Host ein Fehler auftritt, können die logischen Knoten für den fehlerhaften Host auf dem anderen Host erneut gestartet werden.		
DB2_PARTITIONEDLOAD__DEFAULT	Alle unterstützten ESE-Plattformen	Standardwert: YES; Wertebereich: YES/NO
Die Registriervariable 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 ist YES, der angibt, dass in einer ESE-Umgebung, wenn Sie keine ESE-spezifischen LOAD-Optionen angeben, das Laden in allen Partitionen versucht wird, in denen die Zieltabelle definiert ist. Wenn der Wert NO ist, wird das Laden nur in der Partition versucht, mit der das Dienstprogramm LOAD gegenwärtig verbunden ist.		
DB2PORTRANGE	Windows NT	Werte: nnnn:nnnn
Dieser Wert wird auf den TCP/IP-Portbereich gesetzt, der vom FCM verwendet wird, so dass alle zusätzlichen auf einer anderen Maschine erstellten Partitionen den gleichen Portbereich haben.		

Zugehörige Konzepte:

- „DB2-Registriervariablen und DB2-Umgebungsvariablen" auf Seite 565

Variablen des SQL-Compilers

Tabelle 51. Variablen des SQL-Compilers

Variablenname	Betriebssystem	Werte
Beschreibung		
DB2_ANTIJOIN	Alle	Standardwert=NO in ESE-Umgebung Standardwert=YES in anderer als ESE-Umgebung Werte: YES, NO oder EXTEND
In DB2 Universal Database ESE-Umgebungen: Bei Angabe von ″YES″ sucht das Optimierungsprogramm nach Gelegenheiten, „NOT EXISTS”-Unterabfragen in Antiverknüpfungen (Antijoins) umzusetzen, die von DB2 effizienter verarbeitet werden können. In anderen Umgebungen (nicht ESE): Bei Angabe von ″NO″ schränkt das Optimierungsprogramm die Möglichkeiten für die Umsetzung von „NOT EXISTS”-Unterabfragen in Antiverknüpfungen ein. Wenn EXTEND angegeben ist, sucht das Optimierungsprogramm sowohl in ESE- als auch Nicht-ESE-Umgebungen nach Gelegenheiten NOT IN- und NOT EXISTS-Unterabfragen in Antiverknüpfungen umzusetzen.		
DB2_CORRELATED_PREDICATES	Alle	Standardwert=YES Werte: Yes oder No
Der Standardwert für diese Variable ist ″Yes″. Wenn in einer Verknüpfung eindeutige Indizes in korrelierten Spalten vorhanden sind und diese Registriervariable auf ″Yes″ eingestellt ist, versucht das Optimierungsprogramm, die Korrelation der Verknüpfungsvergleichselemente zu erkennen und auszugleichen. Ist diese Registriervariable auf ″Yes″ eingestellt, stellt das Optimierungsprogramm mit Hilfe der KEYCARD-Informationen aus den Statistiken des eindeutigen Index die Fälle von Korrelation fest und passt die kombinierten Selektivitäten der korrelierten Vergleichselemente dynamisch an, wodurch eine genauere Schätzung der Größe und des Aufwands für die Verknüpfung erzielt wird. Auch für die Korrelation einfacher Gleichheitsvergleichselemente wie `WHERE C1=5 AND C2=10` erfolgt eine Anpassung, wenn ein Index für C1 und C2 existiert. Der Index muss nicht eindeutig sein, aber die Spalten der Gleichheitsvergleichselemente müssen alle Spalten des Index abdecken.		
DB2_HASH_JOIN	Alle	Standardwert=YES Werte: YES oder NO
Gibt die Hashverknüpfung als mögliche Verknüpfungsmethode bei der Kompilierung eines Zugriffsplans an.		
DB2_INLIST_TO_NLJN	Alle	Standardwert=NO Werte: YES oder NO

Tabelle 51. Variablen des SQL-Compilers (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		

In einigen Fällen kann der SQL-Compiler ein Vergleichselement mit IN-Liste in eine Verknüpfung 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 eine Verknüpfung über Verschachtelungsschleife mit Hilfe des Index zur Anwendung des Verknüpfungsvergleichselements verknüpft.
 	
Manchmal verfügt das Optimierungsprogramm über keine genauen Informationen, um die beste Verknüpfungsmethode 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 mit Hilfe der Katalogstatistiken ermitteln kann. Diese Registriervariable veranlasst das Optimierungsprogramm, Verknüpfungen über Verschachtelungsschleifen zu bevorzugen, um die Liste von Werten zu verknüpfen und dabei die Tabelle, die die IN-Liste beisteuert, als innere Tabelle in der Verknüpfung zu verwenden.		

DB2_LIKE_VARCHAR	Alle	Standardwert=Y,Y

Tabelle 51. Variablen des SQL-Compilers (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		

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 des API gesteuert.
 	
Diese Registriervariable bestimmt, wie das Optimierungsprogramm ein Vergleichselement der folgenden Form behandelt:
`COLUMN LIKE '%xxxxxx%'`
 	
Dabei ist `xxxxxx` eine beliebige Zeichenfolge.
 	
Die folgende Syntax zeigt, wie diese Registriervariable 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.		

DB2_MINIMIZE_LISTPREFETCH	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 mit Hilfe der Katalogstatistiken ermitteln kann. Diese Registriervariable verhindert, dass das Optimierungsprogramm in solchen Situationen einen Vorablesezugriff über Listen in Betracht zieht.		
DB2_SELECTIVITY	Alle	Standardwert=No Werte: Yes oder No

Tabelle 51. Variablen des SQL-Compilers (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
Diese Registriervariable steuert, wo die Klausel SELECTIVITY in Suchbedingungen in SQL-Anweisungen verwendet werden kann. Wenn diese Registriervariable auf "Yes" gesetzt ist, kann die Klausel SELECTIVITY für die folgenden Vergleichselemente angegeben werden: • Ein einfaches Vergleichselement, in dem mindestens ein Ausdruck Hostvariablen enthält • Ein LIKE-Vergleichselement, in dem Abgleichausdruck (MATCH), der Vergleichselementausdruck oder der Escape-Ausdruck Hostvariablen enthält		
DB2_NEW_CORR_SQ_FF	Alle	Standardwert=OFF Werte: ON oder OFF

Beeinflusst, wenn auf „ON " gesetzt, den Selektivitätswert, der vom SQL-Optimierungsprogramm für bestimmte Vergleichselemente von Unterabfragen ermittelt 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. Zum Beispiel:
`SELECT * FROM T WHERE`
`T.COL = (SELECT MIN(T.COL)`
`FROM T WHERE ...)`		

DB2_PRED_FACTORIZE	Alle	Standardwert=NO Werte: YES oder NO

Gibt an, ob das Optimierungsprogramm nach Gelegenheiten suchen soll, um zusätzliche Vergleichselemente aus Disjunktionen zu extrahieren. In manchen Fällen können die zusätzlichen Vergleichselemente die geschätzte Kardinalität der vorläufigen und endgültigen Ergebnismengen verändern. Mit der folgenden Abfrage:
`SELECT n1.empno,`
`n1.lastname`
`FROM employee n1,`
`employee n2`
`WHERE`
`((n1.lastname='SMITH'`
`AND n2.lastname='JONES')`
`OR (n1.lastname='JONES'`
`AND n2.lastname='SMITH'))`
 	
kann das Optimierungsprogramm die folgenden zusätzlichen Vergleichselemente generieren:
`SELECT n1.empno,`
`n1.lastname`
`FROM employee n1,`
`employee n2`
`WHERE n1.lastname IN`
`('SMITH', 'JONES')`
`AND n2.lastname IN`
`('SMITH', 'JONES')`
`AND`
`((n1.lastname='SMITH'`
`AND n2.lastname='JONES')`
`OR (n1.lastname='JONES'`
`AND n2.lastname='SMITH'))`		

DB2_REDUCED_OPTIMIZATION	Alle	Standardwert=NO Werte: NO, YES, beliebiger INTEGER-Wert, DISABLE

Tabelle 51. Variablen des SQL-Compilers (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
Mit dieser Registriervariablen können Sie entweder reduzierte Optimierungsfunktionen oder 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. **Anmerkung:** Die Optimierungszeit und die Ressourcenbelastung wird zwar reduziert, jedoch erhöht sich das Risiko, dass ein nicht optimaler Datenzugriffsplan generiert wird. Verwenden Sie diese Registriervariable nur, wenn Sie von IBM und 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 ein beliebiger INTEGER-Wert definiert ist Der Effekt ist der gleiche wie bei YES, jedoch mit folgenden zusätzlichen Merkmalen für dynamisch vorbereitete Abfragen, die mit Klasse 5 optimiert werden. Wenn die Gesamtanzahl von Verknüpfungen in einem beliebigen Abfrageblock die Einstellung überschreitet, wechselt das Optimierungsprogramm zur schnellen Verknüpfungsaufzählung (Greedy Join Enumeration), anstatt zusätzliche Optimierungstechniken wie oben 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. Beachten Sie, dass die dynamische Reduzierung der Optimierung in Optimierungsklasse 5 Vorrang vor der Funktionsweise hat, die für eine Optimierungsklasse exakt 5 beschrieben ist, wenn die Variable DB2_REDUCED_OPTIMIZATION auf den Wert YES gesetzt ist, sowie vor der Funktionsweise, die für die INTEGER-Einstellung beschrieben ist.		
DB2_SQLROUTINE_PREPOPTS	Alle	Standardwert=leere Zeichenfolge Werte: • BLOCKING {UNAMBIG | ALL | NO} • DATETIME {DEF | USA | EUR | ISO | JIS | LOC} • DEGREE {1 | *grad-der-parallelität* | ANY} • DYNAMICRULES {BIND | RUN} • EXPLAIN {NO | YES | ALL} • EXPLSNAP {NO | YES | ALL} • FEDERATED {NO | YES} • INSERT {DEF | BUF} • ISOLATION {CS | RR | UR | RS | NC} • QUERYOPT *optimierungsgrad* • VALIDATE {RUN | BIND}

Tabelle 51. Variablen des SQL-Compilers (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
Die Registriervariable DB2_SQLROUTINE_PREPOPTS kann zur Anpassung der Vorkompilierungs- und Bindeoptionen für SQL-Prozeduren verwendet werden.		

Zugehörige Konzepte:

- „Richtlinien für Optimierungsklassen" auf Seite 84
- „Strategien zur Auswahl optimaler Verknüpfungen" auf Seite 188
- „DB2-Registriervariablen und DB2-Umgebungsvariablen" auf Seite 565

Zugehörige Referenzen:

- „Optimierungsklassen" auf Seite 86

Leistungsvariablen

Tabelle 52. Leistungsvariablen

Variablenname	Betriebssystem	Werte
Beschreibung		
DB2AFFINITIES	AIX 5 oder höher, alle Linux-Versionen außer zSeries (32 Bit)	Standardwert = nicht gesetzt Werte: gültiger Pfad zu Konfigurationsdatei

Definiert eine Ressourcenrichtlinie, mit deren Hilfe die von DB2 genutzten Betriebssystemressourcen begrenzt werden können. Unter AIX oder Linux kann diese Registriervariable zum Beispiel zur Begrenzung des von DB2 genutzten Satzes von Prozessoren verwendet werden.
 	
Auf AIX-Maschinen mit aktiviertem NUMA kann eine Richtlinie definiert werden, die angibt, welche Ressourcengruppen (Resource Sets) von DB2 verwendet werden sollen. Wenn die Ressourcengruppenbindung verwendet wird, wird jeder einzelne DB2-Prozess an eine bestimmte Ressourcengruppe gebunden. Dies kann in einigen Leistungsoptimierungsszenarios von Vorteil sein.
 	
Die Registriervariable kann zur Angabe des Pfads zu einer Konfigurationsdatei verwendet werden, die eine Richtlinie zum Binden von DB2-Prozessen an Betriebssystemressourcen definiert. Die Ressourcenrichtlinie ermöglicht Ihnen die Angabe einer Gruppe von Betriebssystemressourcen, auf die DB2 zu beschränken ist. Jeder DB2-Prozess wird an eine einzige Ressource der Gruppe gebunden. Die Ressourcenzuordnung erfolgt in einer zirkulären Reihumverteilung.
 	
Beispiele für Konfigurationsdateien:
`Beispiel 1: Binden aller DB2-Prozesse entweder an CPU 1 oder an CPU 3.`
`<RESOURCE_POLICY>`
`<METHOD>CPU</METHOD>`
`<RESOURCE>1</RESOURCE>`
`<RESOURCE>3</RESOURCE>`
`</RESOURCE_POLICY>`
 	
`Beispiel 2: 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>`
`<METHOD>RSET</METHOD>`
`<RESOURCE>sys/node.03.00000</RESOURCE>`
`<RESOURCE>sys/node.03.00001</RESOURCE>`
`<RESOURCE>sys/node.03.00002</RESOURCE>`
`<RESOURCE>sys/node.03.00003</RESOURCE>`
`</RESOURCE_POLICY>`
 	
Anmerkung: Die Verwendung der Methode RSET setzt die CAP_NUMA_ATTACH-Funktion voraus und wird unter Linux nicht unterstützt.		

DB2_ALLOCATION_SIZE	Alle	Standardwert=8 MB Bereich: 32 KB - 256 MB
Gibt die Größe von Speicherzuordnungen für Pufferpools an. Wenn diese Registriervariable auf einen höheren Wert gesetzt wird, besteht der potenzielle Vorteil darin, dass weniger Zuordnungen erforderlich sind, um eine gewünschte Größe des Speichers zu erreichen, der einem Pufferpool zugeordnet wird. Der potenzielle Nachteil eines höheren Werts für diese Registriervariable 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 MB gesetzt ist und ein Pufferpool um 4 MB verkleinert wird, werden diese 4 MB verschwendet, weil kein ganzes 8-MB-Segment freigegeben werden kann.		
DB2_APM_PERFORMANCE	Alle	Standardwert=OFF Werte: ON, OFF

Tabelle 52. Leistungsvariablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
Setzen Sie diese Variable auf den Wert ON, um die leistungsorientierten Änderungen am Zugriffsplanmanager (APM) zu aktivieren, die sich auf die Funktionsweise des SQL-Cache (Paketcache) auswirken. Diese Einstellungen werden für Produktionssysteme im Normalfall nicht empfohlen. Sie verursachen einige Einschränkungen, wie zum Beispiel die Möglichkeit von Fehlern aufgrund nicht ausreichenden Paketcacheplatzes oder erhöhter Speicherauslastung oder beides. Die Einstellung der Variablen DB2_APM_PERFORMANCE auf den Wert ON aktiviert außerdem den Modus 'Keine Paketsperre' ('No Package Lock'). Dieser Modus ermöglicht den Betrieb des globalen SQL-Cache ohne Verwendung von Paketsperren, bei denen es sich um interne Systemsperren handelt, die Paketeinträge im Cache davor schützen, entfernt zu werden. Der Modus 'Keine Paketsperre' 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	Alle	Standardwert=OFF Werte: ON, OFF
Wenn aktiviert, ermöglicht diese Variable DB2 die Annahme, dass alle Spalten fester Länge, die in einer UPDATE-Anweisung angegeben werden, tatsächlich geändert werden. Dadurch braucht DB2 keinen Vergleich zwischen den vorhandenen Spaltenwerten und den angegebenen neuen Spaltenwerten durchzuführen, um festzustellen, ob sich die Spalte tatsächlich ändert. Wenn bei Verwendung dieser Registriervariablen 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. Diese Registriervariable wird zum Zeitpunkt einer Aktualisierung (UPDATE) geprüft.		
DB2_AVOID_PREFETCH	Alle	Standardwert=OFF Werte: ON oder OFF
Definiert, ob ein Vorablesezugriff bei der Wiederherstellung nach einem Systemabsturz verwendet werden soll. Wenn DB2_AVOID_PREFETCH=ON definiert ist, wird der Vorablesezugriff nicht verwendet.		
DB2_AWE	Windows 2000	Standardwert=null Werte: <eintrag>[;<eintrag>...] mit <eintrag>=<pufferpool-id>,<anzahl physischer seiten>, <anzahl von adressfenstern>
Ermöglicht DB2 UDB auf 32-Bit-Plattformen unter Windows 2000 die Zuordnung von Pufferpools, die bis zu 64 GB Speicher belegen. Die Unterstützung von AWE-Pufferpools (AWE - Address Windowing Extensions) erfordert eine korrekte Konfiguration von Windows 2000. Dazu gehören die Zuordnung der Berechtigung „Seiten im Speicher sperren" für den Benutzer, die Zuordnung der physischen Seiten und der Adressfensterseiten sowie die Einstellung dieser Registriervariablen. Zur Einstellung dieser Variablen benötigen Sie die Pufferpool-ID des Pufferpools, der für die AWE-Unterstützung verwendet werden soll. Die ID des Pufferpools können Sie der Spalte BUFFERPOOLID in der Systemkatalogsicht SYSCAT.BUFFERPOOLS entnehmen. **Anmerkung:** • Wenn die AWE-Unterstützung aktiviert ist, kann für keinen der Pufferpools in der Datenbank erweiterter Speicher verwendet werden. • Pufferpools, auf die diese Registriervariable verweist, müssen bereits in SYSCAT.SYSBUFFERPOOLS vorhanden sein. • Pufferpools, die für AWE aktiviert sind, haben Vorrang vor Pufferpools, die für blockbasierte E/A aktiviert sind. Wenn ein Pufferpool für AWE und blockbasierte E/A konfiguriert ist, hat AWE Vorrang vor der blockbasierten E/A.		
DB2_BINSORT	Alle	Standardwert=YES Werte: YES oder NO

Tabelle 52. Leistungsvariablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
Aktiviert einen neuen Sortieralgorithmus, der die CPU-Zeit und die abgelaufene Zeit für Sortierungen reduziert. Dieser neue Algorithmus weitet die extrem effiziente Integer-Sortiertechnik von DB2 UDB auf alle Sortierdatentypen wie BIGINT, CHAR, VARCHAR, FLOAT und DECIMAL sowie auf Kombinationen aus diesen Datentypen aus. Zur Aktivierung dieses neuen Algorithmus verwenden Sie den folgenden Befehl: `db2set DB2_BINSORT = yes`		
DB2BPVARS	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 Behältern verwenden sollen. Die andere Gruppe von Parametern, die auf allen Plattformen verfügbar ist, beeinflusst die Funktionsweise des Vorablesezugriffs.
 	
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 folgenden Zeilen enthalten:
 	
`NO_NT_SCATTER = 1`
`NUMPREFETCHQUEUES = 2`
 	
Wenn die Datei `bpvars.vars` zum Beispiel im Pfad F:\vars\ gespeichert ist, geben Sie zur Definition dieser Variablen den folgenden Befehl ein:
`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 Behälter empfohlen, für die Sie bereits den Parameter DB2NTNOCACHE auf den Wert 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 Behälter explizit auszuschließen. Bestimmte Parameter werden dazu verwendet, den SCATTER-Lesezugriff für alle Behälter des angegebenen Typs zu aktivieren. Für jeden dieser Parameter ist der Standardwert 0 (bzw. OFF). Die möglichen Werte sind: 0 (oder OFF) und 1 (oder ON).
Anmerkung: Sie können den SCATTER-Lesezugriff nur aktivieren, wenn DB2NTNOCACHE auf den ON gesetzt ist, um die Windows-Dateicachefunktion abzuschalten. Wenn DB2NTNOCACHE auf den Wert OFF gesetzt oder nicht definiert ist, wird eine Warnung in das Benachrichtigungsprotokoll für die Systemverwaltung geschrieben, wenn Sie versuchen, den SCATTER-Lesezugriff für einen Behälter zu aktivieren. Der SCATTER-Lesezugriff bleibt in diesem Fall inaktiviert.		

Tabelle 52. Leistungsvariablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
Parameter zur Anpassung des Vorablesezugriffs Die Parameter zur Anpassung des Vorablesezugriffs heißen NUMPREFETCHQUEUES und PREFETCHQUEUESIZE. Diese Parameter stehen auf allen Plattformen zur Verfügung und können zur Verbesserung des Datenvorabzugriffs für Pufferpools verwendet werden. Betrachten Sie zum Beispiel den sequenziellen Vorablesezugriff, bei dem der gewünschte PREFETCHSIZE-Wert in PREFETCHSIZE/EXTENTSIZE Vorableseanforderung dividiert wird. In diesem Fall werden die Anforderungen in Vorablesewarteschlangen gestellt, aus denen E/A-Server zugeteilt werden, um asynchrone E/A-Operationen auszuführen. Standardmäßig verwaltet DB2 eine Warteschlange in der Größe max(100 , 2*NUM_IOSERVERS) für jede Datenbankpartition. In einigen Umgebungen lässt sich die Leistung entweder durch mehr Warteschlangen oder durch Warteschlangen einer anderen Größe oder beides verbessern. Die Anzahl der Warteschlangen für den Vorablesezugriff sollte höchstens die Hälfte der Anzahl von E/A-Servern betragen. Berücksichtigen Sie bei der Einstellung dieser Parameter andere Parameter wie beispielsweise PREFETCHSIZE, EXTENTSIZE, NUM_IOSERVERS und die Pufferpoolgröße sowie die Merkmale der Auslastung wie beispielsweise die Anzahl gleichzeitig verbundener Benutzer. Wenn Sie meinen, dass die Standardwerte für Ihre Umgebung zu klein sind, erhöhen Sie die Werte zunächst nur leicht. Zum Beispiel könnten Sie NUMPREFETCHQUEUES=4 und PREFETCHQUEUESIZE=200 definieren. Nehmen Sie Änderungen an diesen Parametern kontrolliert vor, so dass Sie die Wirkungen der Änderung überwachen und bewerten können. Für NUMPREFETCHQUEUES ist der Standardwert 1, wobei der Wertebereich von 1 bis NUM_IOSERVERS geht. Wenn Sie NUMPREFETCHQUEUES auf einen kleineren Wert als 1 setzen, wird er auf den Wert 1 angepasst. Wenn Sie ihn auf einen höheren Wert als NUM_IOSERVERS setzen, wird er auf den Wert von NUM_IOSERVERS angepasst. Der Standardwert für PREFETCHQUEUESIZE ist max(100,2*NUM_IOSERVERS). Der Wertebereich geht von 1 bis 32767. Wenn Sie PREFETCHQUEUESIZE auf einen kleineren Wert als 1 setzen, wird er auf den Standardwert angepasst. Wenn er auf einen höheren Wert als 32767 gesetzt wird, wird er auf 32767 angepasst.		
DB2CHKPTR	Alle	Standardwert=OFF Werte: ON oder OFF
Definiert, ob die Zeigerprüfung für Eingaben erforderlich ist oder nicht.		
DB2CHKSQLDA	Alle	Standardwert=OFF Werte: ON oder OFF
Definiert, ob die Prüfung von SQL-Deskriptorbereichen (SQLDA) für Eingaben erforderlich ist oder nicht.		
DB2_ENABLE_BUFPD	Alle	Standardwert=YES Werte: ON oder OFF
Gibt an, ob DB2 eine Zwischenpufferung zur Verbesserung der Abfrageleistung einsetzt. Die Pufferung führt möglicherweise nicht in allen Umgebungen zu einer Verbesserung der Abfrageleistung. Um einzelne Verbesserungen in der Abfrageleistung zu ermitteln, sollten Tests durchgeführt werden.		
DB2_EVALUNCOMMITTED	Alle	Standardwert=OFF Werte: ON, OFF

Tabelle 52. Leistungsvariablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
Wenn aktiviert, ermöglicht diese Variable, dass bei Tabellen- oder Indexsuchzugriffen Zeilensperren nach Möglichkeit verzögert bzw. vermieden werden, bis ein Datensatz ermittelt wird, der die Auswertung der Vergleichselemente erfüllt. Wenn diese Variable aktiviert ist, kann eine Auswertung von Vergleichselementen für nicht festgeschriebene Daten stattfinden. Dies gilt nur für Anweisungen, die mit der Isolationsstufe Cursorstabilität oder Lesestabilität arbeiten. Bei Indexsuchen muss es sich um einen Index des Typs 2 handeln. Darüber hinaus werden beim Tabellensuchenzugriff gelöschte Zeilen bedingungslos übersprungen, während gelöschte Schlüssel beim Durchsuchen von Indizes des Typs 2 nicht übersprungen werden, sofern nicht die Registriervariable DB2_SKIPDELETED ebenfalls definiert ist. Die Aktivierung der Registriervariablen DB2_EVALUNCOMMITTED wird bei der Ausführung des Befehls **db2start** wirksam, während die Entscheidung, ob ein verzögertes Sperren anwendbar ist, beim Kompilieren oder Binden der Anweisung getroffen wird.		
DB2_EXTENDED_OPTIMIZATION	Alle	Standardwert=OFF Werte: ON oder OFF
Gibt an, ob das Abfrageoptimierungsprogramm die Optimierungserweiterungen zum Verbessern der Abfrageleistung verwendet. Die Erweiterungen verbessern die Abfrageleistung möglicherweise nicht in allen Umgebungen. Um einzelne Verbesserungen in der Abfrageleistung zu ermitteln, sollten Tests durchgeführt werden.		
DB2_KEEPTABLELOCK	Alle	Standardwert=OFF Werte: ON, OFF
Wenn aktiviert, ermöglicht diese Variable DB2, die Freigabe der Tabellensperre zu vermeiden, wenn eine Isolationsstufe UR (Nicht festgeschriebener Lesevorgang) oder CS (Cursorstabilität) geschlossen wird. Die Tabellensperre, die behalten wird, wird am Ende der Transaktion freigegeben, wie dies bei Suchen mit der Isolationsstufe RS (Lesestabilität) und RR (Wiederholtes Lesen) der Fall ist. Diese Registriervariable wird zum Zeitpunkt des Kompilierens oder Bindens einer Anweisung geprüft.		
DB2_LGPAGE_BP	Nur AIX 5.x (64-Bit) Linux	Standardwert=OFF Werte: ON oder OFF

Tabelle 52. Leistungsvariablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		

Die Registriervariable DB2_LGPAGE_BP wird verwendet, um unter AIX 5.x oder für eine beliebige Linux-Architektur mit der entsprechenden Kernelunterstützung die Unterstützung für große Seiten zu aktivieren. Die Variable wird lediglich für DB2 UDB für AIX (64-Bit-Edition) und DB2 UDB für Linux unterstützt. Die Verwendung großer Seiten dient hauptsächlich zur Leistungsverbesserung hochleistungsfähiger Datenverarbeitungsanwendungen. Bei Anwendungen, die einen intensiven Speicherzugriff erfordern und große Mengen an virtuellem Speicher verwenden, kann die Verwendung umfangreicher Seiten zu einer Leistungsverbesserung führen. Um DB2 für die Verwendung umfangreicher Seiten zu aktivieren, müssen Sie zunächst das Betriebssystem für die Verwendung dieser Seiten konfigurieren.
 	
Unter 64-Bit-DB2 für AIX reduziert diese Variable die Größe des gemeinsamen Speichersegments, das den Datenbankspeicher unterstützt, auf den Mindestbedarf (standardmäßig wird ein 64-GB-Segment erstellt; weitere Informationen finden Sie unter dem Konfigurationsparameter 'database_memory'). Dies soll vermeiden, dass mehr Speicher im RAM belegt wird, als wahrscheinlich genutzt wird.
 	
Die Aktivierung dieser Variablen schränkt die Fähigkeit ein, die Konfiguration für den gemeinsamen Datenbankspeicher insgesamt zu erhöhen, zum Beispiel, um die Pufferpools zu vergrößern.
 	
Unter Linux besteht die zusätzliche Voraussetzung, dass die Bibliothek libcap.so verfügbar sein muss. Diese Bibliothek muss installiert sein, damit diese Option funktioniert. Wenn diese Option aktiviert wird und die Bibliothek nicht im System vorhanden ist, inaktiviert DB2 die großen Kernelseiten und setzt die Verarbeitung wie zuvor fort.
 	
Um unter Linux zu prüfen, ob große Kernelseiten (Large Kernel Pages) verfügbar sind, können Sie den folgenden Befehl ausführen:
`cat /proc/meminfo`
 	
Wenn sie verfügbar sind, sollten die drei folgenden Zeilen angezeigt werden (mit anderen Werten, je nach Größe des auf Ihrer Maschine konfigurierten Speichers):
`HugePages_Total: 200`
`HugePages_Free: 200`
`Hugepagesize: 16384 kB`
 	
Wenn Ihnen diese Zeilen nicht angezeigt werden oder wenn HugePages_Total den Wert `0` hat, ist eine Konfiguration des Betriebssystems oder des Kernels erforderlich.		

DB2MAXFSCRSEARCH	Alle	Standardwert=5 Werte: -1, 1 bis 33 554
Gibt an, wie viele Steuersätze für freien Speicherbereich beim Hinzufügen eines Eintrags zu einer Tabelle durchsucht werden. Standardmäßig werden fünf Steuersätze für freien Speicherbereich durchsucht. Mit diesem Wert können Sie die Gewichtung zwischen Einfügegeschwindigkeit und Speicherwiederverwendung verändern. 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_NON_TABLE_LOCKS	Alle	Standardwert=YES Werte: siehe unten.

Tabelle 52. Leistungsvariablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
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 behalten werden, selbst wenn die Transaktion ihre Verwendung beendet hat. Da Transaktionen häufig mehrmals auf die gleiche Tabelle zugreifen, kann das Behalten der Sperren und das Ändern ihres Status in NON die Leistung verbessern, da diese Sperren nicht neu erstellt werden müssen. Im Hinblick auf optimale Ergebnisse ist der für diese Variable empfohlene Wert die maximale Anzahl von Tabellen, auf die durch Verbindungen voraussichtlich zugegriffen wird. Wenn kein benutzerdefinierter Wert angegeben wird, ist der Standardwert folgender: Wenn die Sperrenliste größer oder gleich SQLP_THRESHOLD_VAL_OF_LRG_LOCKLIST_SZ_FOR_MAX_NON_LOCKS (zurzeit 8000) ist, ist der Standardwert gleich SQLP_DEFAULT_MAX_NON_TABLE_LOCKS_LARGE (zurzeit 150). Ansonsten ist der Standardwert SQLP_DEFAULT_MAX_NON_TABLE_LOCKS_SMALL (zurzeit 0).		
DB2MEMDISCLAIM	AIX	Standardwert=YES Werte: YES oder NO
Unter AIX verfügt der von DB2-Prozessen verwendete Speicher u. U. 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 den (einstellbaren) Richtlinien für die Speicherzuordnung im Rahmen der Verwaltung des virtuellen Speichers auf dem AIX-System ab. Die Registriervariable DB2MEMDISCLAIM steuert, ob DB2-Agenten AIX ausdrücklich auffordern, die Zuordnung des reservierten Paging-Bereichs zu dem freigegebenen Speicher aufzuheben. 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.		
DB2MEMMAXFREE	Alle	Standardwert = 8 388 608 Byte Werte: 0 bis 2^{32}-1 Byte
Gibt die maximale Anzahl von Byte an ungenutztem privaten Speicher an, der von DB2-Prozessen zurückbehalten wird, bevor ungenutzter Speicher an das Betriebssystem zurückgegeben wird.		
DB2_MMAP_READ	AIX	Standardwert=ON Werte: ON oder OFF
Wird in Verbindung mit DB2_MMAP_WRITE verwendet, damit DB2 mmap als alternative E/A-Methode verwenden kann. In den meisten Umgebungen sollte mmap verwendet werden, um Sperren des Betriebssystems zu vermeiden, wenn mehrere Prozesse parallel in verschiedene Abschnitte derselben Datei schreiben. Werden diese Variablen auf ON eingestellt, umgehen Daten, die in die DB2-Pufferpools gelesen und aus diesen geschrieben werden, den AIX-Speichercache. Wenn der DB2-Pufferpool bereits relativ klein ist und Sie diesen Pufferpool nicht vergrößern können oder wollen, sollten Sie die Nutzung des AIX-Speichercaches in Erwägung ziehen, indem Sie die Variablen DB2_MMAP_READ und DB2_MMAP_WRITE auf OFF setzen.		
DB2_MMAP_WRITE	AIX	Standardwert=ON Werte: ON oder OFF

Tabelle 52. Leistungsvariablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
Wird in Verbindung mit DB2_MMAP_READ verwendet, damit DB2 mmap als alternative E/A-Methode verwenden kann. In den meisten Umgebungen sollte mmap verwendet werden, um Sperren des Betriebssystems zu vermeiden, wenn mehrere Prozesse parallel in verschiedene Abschnitte derselben Datei schreiben. Werden diese Variablen auf ON eingestellt, umgehen Daten, die in die DB2-Pufferpools gelesen und aus diesen geschrieben werden, den AIX-Speichercache. Wenn der DB2-Pufferpool bereits relativ klein ist und Sie diesen Pufferpool nicht vergrößern können oder wollen, sollten Sie die Nutzung des AIX-Speichercaches in Erwägung ziehen, indem Sie die Variablen DB2_MMAP_READ und DB2_MMAP_WRITE auf OFF setzen.		
DB2_NO_FORK_CHECK	UNIX	Standardwert=OFF Werte: ON, OFF
Wenn aktiviert, 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.		
DB2_NO_MPFA_FOR_NEW_DB	Alle	Standardwert=nicht gesetzt Werte: YES
Für Datenbanken, die durch den Befehl CREATE DATABASE oder die äquivalente API erstellt werden, ist die mehrseitige Dateizuordnung (MPFA) aktiviert. Wenn die MPFA für eine Datenbank aktiviert ist, kann sie nicht mehr inaktiviert werden. Zur Erstellung einer Datenbank mit inaktivierter MPFA setzen Sie diese Registriervariable auf YES und starten das Exemplar vor der Erstellung der Datenbank erneut. Wenn diese Registriervariable definiert ist, werden alle Datenbanken mit inaktivierter MPFA erstellt. Zur Aktivierung der MPFA für eine Datenbank mit inaktivierter MPFA verwenden Sie den Befehl **db2empfa**.		
DB2NTMEMSIZE	Windows NT	Standardwert =(je nach Speichersegment unterschiedlich)
Windows NT erfordert, dass alle gemeinsam benutzten 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 NT (falls erforderlich). In den meisten Situationen sollte der Standardwert ausreichen. Die Angaben für Speichersegmente, Standardgrößen und Überschreibungsoptionen lauten wie folgt: 1) Datenbankkernel: Standardgröße ist 16777216 (16 MB); Überschreibungsoption ist DBMS:<anzahl_byte>. 2) Parallele FCM-Puffer: Standardgröße ist 22020096 (21 MB); Überschreibungsoption ist FCM:<anzahl_byte>. 3) GUI für Datenbankverwaltung: Standardgröße ist 33554432 (32 MB); Überschreibungsoption ist DBAT:<anzahl_byte>. 4) Abgeschirmte gespeicherte Prozeduren: Standardgröße ist 16777216 (16 MB); Überschreibungsoption ist APLD:<anzahl_byte> Sie können mehrere Segmente überschreiben, indem Sie die Überschreibungsoptionen durch ein Semikolon (;) voneinander trennen. Wenn Sie zum Beispiel den Datenbankkernel auf ungefähr 256 KB und die FCM-Puffer auf ungefähr 64 MB begrenzen wollen, geben Sie Folgendes an: `db2set DB2NTMEMSIZE=DBMS:256000;FCM:64000000`		
DB2NTNOCACHE	Windows NT	Standardwert=OFF Wert: ON oder OFF
Gibt an, ob DB2 Datenbankdateien mit der Option NOCACHE öffnet. Wenn DB2NTNOCACHE=ON definiert ist, wird das Zwischenspeichern im Dateisystemcache inaktiviert. Wenn DB2NTNOCACHE=OFF definiert 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, so dass der Pufferpool oder der Sortierspeicher vergrößert werden können. Unter Windows NT werden Dateien im Cache zwischengespeichert, wenn sie geöffnet werden. Dies ist die Standardfunktionsweise. 1 MB wird aus einem Systempool für jedes GB in der Datei reserviert. Verwenden Sie diese Registriervariable, 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.		

Tabelle 52. Leistungsvariablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
DB2NTPRICLASS	Windows NT	Standardwert=null Wert: R, H, (beliebiger anderer Wert)
Stellt die Prioritätsklasse für das DB2-Exemplar ein (Programm DB2SYSCS.EXE). Es gibt drei Prioritätsklassen: • NORMAL_PRIORITY_CLASS (die Standardprioritätsklasse) • REALTIME_PRIORITY_CLASS (Wert „R") • HIGH_PRIORITY_CLASS (Wert „H") 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:** Bei Verwendung dieser Variablen ist Sorgfalt geboten. Eine falsche Verwendung kann sich negativ auf die Gesamtleistung des Systems auswirken. Weitere Informationen finden Sie bei der API *SetPriorityClass()* in der Win32-Dokumentation.		
DB2NTWORKSET	Windows NT	Standardwert=1,1
Dient zur Änderung der minimalen und der maximalen Größe des Arbeitsbereichs, der für DB2 verfügbar ist. Standardmäßig kann der Arbeitsbereich eines Prozesses, wenn unter Windows NT keine Seitenwechsel auftreten, nach Bedarf wachsen. Wenn jedoch Seitenwechsel auftreten, liegt der maximale Arbeitsbereich, den ein Prozess haben kann, bei annähernd 1 MB. Mit DB2NTWORKSET kann dieser Standardwert überschrieben werden. Geben Sie DB2NTWORKSET für DB2 in der Syntax DB2NTWORKSET=`min,max` an, wobei min und max MB-Werte sind.		
DB2_OVERRIDE_BPF	Alle	Standardwert = nicht gesetzt Werte: eine positive Anzahl von Seiten ODER <eintrag>[;<eintrag>...], wobei <eintrag>=<pufferpool-ID>,<anzahl der seiten> ist.
Gibt die Größe des Pufferpools in Seiten an, der bei der Aktivierung der Datenbank oder bei der Herstellung der ersten Verbindung zu erstellen ist. Diese Variable ist nützlich, wenn es während der Datenbankaktivierung oder der ersten Verbindung aufgrund begrenzter Speicherkapazitäten zu Fehlern kommt. Wenn selbst ein Pufferpool der Minimalgröße von 16 Seiten vom Datenbankmanager nicht bereitgestellt werden kann, hat der Benutzer die Möglichkeit, mit Hilfe dieser Umgebungsvariablen eine kleinere Anzahl von Seiten anzugeben und den Versuch zu wiederholen. Die Speicherknappheit kann infolge des seltenen Falls eines realen Speichermangels auftreten oder durch den Versuch seitens des Datenbankmanagers verursacht werden, Speicher für große, nicht korrekt konfigurierte Pufferpools zuzuordnen. Wenn dieser Wert gesetzt wird, überschreibt er 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, so dass sie gestartet werden können.		
DB2_PINNED_BP	AIX, HP-UX	Standardwert=NO Werte: YES oder NO

Tabelle 52. Leistungsvariablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		

Mit dieser Variable wird der globale Datenbankspeicher (einschließlich Pufferpools), der der Datenbank im Hauptspeicher zugeordnet ist, unter einigen AIX-Betriebssystemen angegeben. Wenn dieser globale Datenbankspeicher im Hauptspeicher des Systems bleibt, ist eine konsistentere Datenbankleistung möglich.
 	
Zum Beispiel führt eine Auslagerung des Pufferpools aus dem Systemhauptspeicher zu einer verminderten Datenbankleistung führen. Die geringere Zahl von Lese- und Schreibvorgängen auf der Festplatte, wenn sich die Pufferpools im Systemspeicher befinden, verbessert die Datenbankleistung. Falls andere Anwendungen mehr Hauptspeicher benötigen, sollten Sie abhängig vom Systemhauptspeicherbedarf die Auslagerung des globalen Datenbankspeichers aus dem Hauptspeicher zulassen.
 	
Unter 64-Bit-DB2 für AIX reduziert diese Variable die Größe des gemeinsamen Speichersegments, das den Datenbankspeicher unterstützt, auf den Mindestbedarf (standardmäßig wird ein 64-GB-Segment erstellt; weitere Informationen finden Sie unter dem Konfigurationsparameter 'database_memory'). Dies soll vermeiden, dass mehr Speicher im RAM belegt wird, als wahrscheinlich genutzt wird.
 	
Die Aktivierung dieser Variablen schränkt die Fähigkeit ein, die Konfiguration für den gemeinsamen Datenbankspeicher insgesamt zu erhöhen, zum Beispiel, um die Pufferpools zu vergrößern.
 	
Für HP-UX in einer 64-Bit-Umgebung müssen Sie zusätzlich zur Änderung dieser Registriervariablen der DB2-Exemplargruppe das Zugriffrecht MLOCK erteilen. Dazu führt ein Benutzer mit Rootberechtigung die folgenden Aktionen aus:
1. Er fügt die DB2-Exemplargruppe der Datei `/etc/privgroup` hinzu. Wenn die DB-Exemplargruppe beispielsweise zur Gruppe `db2iadm1` gehört, müssen Sie der Datei `/etc/privgroup` die folgende Zeile hinzufügen:
`db2iadm1 MLOCK`
2. Führen Sie den folgenden Befehl aus:
`setprivgrp -f /etc/privgroup`		

DB2PRIORITIES	Alle	Einstellung der Werte von der Plattform abhängig
Steuert die Prioritäten von DB2-Prozessen und Threads.		
DB2_SCATTERED_IO	Linux	Standardwert=OFF Werte: ON, OFF
Diese Variable aktiviert die SCATTER-E/A, die readv() zum Lesen von der Festplatte verwendet. Wenn Sie mit einem System arbeiten, das die Linux-Kernelkorrektur zur Leistungsverbesserung bei der über einen Vektor definierten unformatierten E/A enthält, sollten Sie diese Variable aktivieren, um die Leistung zu erhöhen. Diese Kernelkorrektur (Patch) ist zurzeit UnitedLinux 1.0 SP2 oder eine höhere für IA-32 und wird in allen künftigen Linux 2.6-Kerneln enthalten sein.		
DB2_SKIPDELETED	Alle	Standardwert=OFF Werte: ON, OFF
Wenn aktiviert, ermöglicht diese Registriervariable 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 festgeschriebene Pseudolöschungen von Schlüsseln in Indizes des Typs 2 werden jedoch nur übersprungen, wenn auch DB2_SKIPDELETED aktiviert ist. Diese Registriervariable wirkt sich nicht auf das Verhalten von Cursorn in den DB2-Katalogtabellen aus. Die Aktivierung dieser Registriervariablen wird mit der Ausführung von db2start wirksam.		

Tabelle 52. Leistungsvariablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
DB2_SMS_TRUNC_TMPTABLE_THRESH	Windows	-1, 0-n, wobei n=Anzahl der EXTENTSIZE-Bereiche pro Behälter, die zu behalten sind
Gibt einen minimalen Schwellenwert für die Dateigröße an, ab dem die Datei, die eine temporäre Tabelle darstellt, in SMS-Tabellenbereichen behalten wird. Durch Einstellen dieser Variablen auf einen Wert größer 0 kann der Systemaufwand verringert werden, der mit dem Löschen und erneuten Erstellen der Datei bei jeder Verwendung einer temporären Tabelle verbunden ist. Standardmäßig wird die Datei für eine temporäre Tabelle, die nicht mehr benötigt wird, auf eine EXTENTSIZE-Größe pro Behälter abgeschnitten. Wenn die Datei bereits einen EXTENTSIZE-Wert groß oder kleiner ist, bleibt sie, wie sie ist. Wenn der Wert dieser Variablen größer als 1 ist, wird eine größere Datei behalten. Wenn diese Variable auf den Wert -1 gesetzt wird, wird die Datei überhaupt nicht abgeschnitten und kann uneingeschränkt anwachsen, soweit die Systemressourcen dies zulassen. Wenn diese Variable auf den Wert 0 gesetzt wird, erfolgt keine spezielle Schwellenwertberechnung. Stattdessen wird die Datei auf 0 abgeschnitten, wenn die Tabelle nicht mehr benötigt wird.		
DB2_SORT_AFTER_TQ	Alle	Standardwert=NO Werte: YES oder NO
Gibt an, wie das Optimierungsprogramm mit gezielt übertragenen Tabellenwarteschlangen in einer partitionierten Datenbank 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 in der Regel auf der sendenden Seite und fügt die Zeilen auf der empfangenden Seite zusammen. Wenn DB2_SORT_AFTER_TQ= `YES` ist, überträgt das Optimierungsprogramm in der Regel die Zeilen unsortiert, fügt sie nicht auf der empfangenden Seite zusammen und sortiert die Zeilen auf der empfangenden Seite, nachdem alle Zeilen empfangen wurden.		
DB2_SELUDI_COMM_BUFFER	Alle	Standardwert=OFF Werte=ON, OFF
Gilt für die Verarbeitung von Blockcursorn über Abfragen mit SELECT from UPDATE/INSERT/DELETE (UDI). Wenn aktiviert, verhindert diese Registriervariable, dass das Ergebnis einer Abfrage in einer temporären Tabelle gespeichert wird. Stattdessen versucht DB2 bei der OPEN-Verarbeitung eines Blockcursors für eine SELECT from UDI-Abfrage, das gesamte Ergebnis der Abfrage direkt im Speicherbereich der Kommunikationspuffer zu puffern. **Anmerkung:** Wenn der Kommunikationspufferbereich für das gesamte Ergebnis der Abfrage nicht ausreicht, wird ein SQLCODE-Wert -906 ausgegeben und die Transaktion 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 Registriervariable wird in partitionierten Datenbankumgebungen oder bei aktivierter partitionsinterner Parallelität nicht unterstützt.		
DB2_TRUSTED_BINDIN	Alle	Standardwert=OFF Werte=OFF, ON, CHECK

Tabelle 52. Leistungsvariablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
Wenn die Variable DB2_TRUSTED_BINDIN aktiviert ist, beschleunigt sie die Ausführung von SQL-Anweisungen, 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-Anweisungen statt, die in einer eingebetteten, nicht abgeschirmten gespeicherten Prozedur enthalten sind. Dies führt zu einer beschleunigten Verarbeitung eingebetteter SQL-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_DATALINK • 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 angetroffen werden, wird ein SQLCODE-Wert –804, SQLSTATE-Wert 07002 zurückgegeben. **Anmerkung:** Der Datentyp und die Länge der Eingabehostvariable 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.		
DB2_USE_ALTERNATE_PAGE_CLEANING	Alle	Standardwert = nicht gesetzt Werte: ON, OFF
Gibt an, ob DB2 die alternative Methode der Seitenbereinigungsalgorithmen anstelle der Standardmethode der Seitenlöschfunktionen verwendet. Wenn diese Variable auf den Wert ON gesetzt ist, verwendet DB2 eine proaktive Methode zur Seitenbereinigung, 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 Platten-E/A-Bandbreite 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.		

Zugehörige Konzepte:

• „DB2-Registriervariablen und DB2-Umgebungsvariablen" auf Seite 565

Data Links-Variablen

Tabelle 53. Data Links-Variablen

Variablenname	Betriebssystem	Werte
Beschreibung		
DLFM_BACKUP_DIR_NAME	AIX, Windows NT, Windows 2000, Solaris-Betriebsumgebung	Standardwert: Null Werte: ein beliebiger gültiger Pfad
Gibt den Pfad des Verzeichnisses an, in dem archivierte Dateien gesichert werden, wenn DLFM_BACKUP_TARGET auf LOCAL eingestellt ist.		
DLFM_BACKUP_TARGET	AIX, Windows NT, Windows 2000, Solaris-Betriebsumgebung	Standardwert: Null Werte: LOCAL, TSM, XBSA
Gibt den zu verwendenden Sicherungsdatenträger an. Wird diese Variable auf LOCAL eingestellt, muss die Variable DLFM_BACKUP_DIR_NAME definiert werden. Wird diese Variable auf XBSA eingestellt, muss die Variable DLFM_BACKUP_TARGET_LIBRARY definiert werden. Wenn Sie die Einstellung dieser Registrierdatenbankvariablen von einem Ziel in ein anderes ändern, werden die archivierten Dateien nicht verschoben. Nur neue Sicherungen werden an die neue Position platziert. Zuvor archivierte Dateien werden nicht verschoben.		
DLFM_BACKUP_TARGET_LIBRARY	AIX, Windows NT, Windows 2000, Solaris-Betriebsumgebung	Standardwert: Null Werte: ein beliebiger gültiger Pfad zum Namen der DLL oder der gemeinsam benutzten Bibliothek
Gibt den vollständig qualifizierten Pfad zur DLL des XBSA-kompatiblen Archivierungsserver oder zur gemeinsam benutzten Bibliothek an. Diese Bibliothek wird unter Verwendung der Bibliothek *libdfmxbsa.a* geladen. Diese Variable muss definiert werden, wenn DLFM_BACKUP_TARGET auf XBSA eingestellt ist. Diese Variable gilt nicht, wenn DLFM_BACKUP_TARGET auf einen anderen Wert eingestellt wird.		
DLFM_GC_MODE	AIX, Windows NT, Windows 2000, Solaris-Betriebsumgebung	Standardwert: PASSIVE Werte: SLEEP, PASSIVE oder ACTIVE
Gibt die Steuerung der Bereinigung des Speichers von ungenutzten Dateien (Garbage File Collection) auf dem Data Links-Server an. Beim Wert SLEEP erfolgt keine Bereinigung. Wenn auf den Wert PASSIVE gesetzt, wird die Bereinigung nur dann ausgeführt, wenn keine anderen Transaktionen aktiv sind. Wenn auf ACTIVE gesetzt, wird die Speicherbereinigung durchgeführt, auch wenn andere Anwendungen aktiv sind.		
DLFM_INSTALL_PATH	AIX, Windows NT, Windows 2000, Solaris-Betriebsumgebung	Standardwert Unter AIX und der Solaris-Betriebsumgebung: `/home/<exemplar>/sqllib/bin`, wobei <exemplar> die Exemplar-ID von Data Links Manager ist. Unter Windows: `%DB2PATH%\bin` (sofern %DB2PATH% definiert ist) oder `c:\sqllib\bin` (sofern %DB2PATH% nicht definiert ist) Bereich: beliebiger gültiger Pfad
Gibt den Pfad an, in dem die ausführbaren Data Links-Dateien installiert sind.		

Tabelle 53. Data Links-Variablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
DLFM_PORT	AIX, Windows NT, Windows 2000, Solaris-Betriebsumgebung	Standardwert: 50100 Werte: jede gültige Portnummer
Gibt die Portnummer an, die zur Kommunikation mit dem Data Links-Server verwendet wird, auf dem DB2 Data Links Manager aktiv ist.		
DLFM_TSM_MGMTCLASS	AIX, Windows NT, Windows 2000, Solaris-Betriebsumgebung	Standardwert: die Standard-TSM-Verwaltungsklasse Werte: jede gültige TSM-Verwaltungsklasse
Gibt an, welche TSM-Verwaltungsklasse zum Archivieren und Abrufen von verknüpften Dateien verwendet wird. Wenn für diese Variable kein Wert definiert ist, wird die TSM-Standardverwaltungsklasse verwendet.		
DLFM_START_ASNCOPYD	AIX, Windows NT, Windows 2000, Solaris-Betriebsumgebung	Standardwert: NO Werte: YES, NO
Gibt an, ob der Data Links Manager-Replikationsdämon (DLFM_ASNCOPYD) immer beim Start von DLFM gestartet wird. Der Data Links Manager-Replikationsdämon muss gestartet werden, wenn DB2 Replication verwendet wird, um DATALINK-Dateien zum bzw. vom Data Links Manager-Server zu kopieren. Wird diese Variable auf YES eingestellt, muss DLFM_ASNCOPYD_PORT definiert werden.		
DLFM_ASNCOPYD_PORT	AIX, Windows NT, Windows 2000, Solaris-Betriebsumgebung	Standardwert: Null Werte: jede gültige Portnummer
Gibt die Nummer des TCP/IP-Ports an, an dem der Data Links Manager-Replikationsdämon (DLFM_ASNCOPYD) Anforderungen für Dateireplikationen empfängt. Diese Variable muss definiert werden, wenn die Variable DLFM_START_ASNCOPYD auf YES eingestellt ist.		
DLFM_NUM_ARCHIVE_SUBSYSTEMS	AIX, Windows NT, Windows 2000, Solaris-Betriebsumgebung	Standardwert: 2 Werte: alle Zahlen größer-gleich 1.
Gibt die Anzahl der DLFM Copy Daemon-Prozesse an, die auf einem bestimmten DLFM-Server ausgeführt werden sollen. Je größer die Anzahl der Kopierprozesse, desto größer der Durchsatz beim Sichern verbundener Dateien. Allerdings sollte dieser Wert der Anzahl der E/A-Kanäle entsprechen, die zum Kopieren verbundener Dateien in den ausgewiesenen Archivierungsbereich verfügbar sind. Ist der Wert zu hoch, kann die Menge der verbrauchten Systemressourcen den Nutzen der E/A-Parallelität mindern.		
DLFM_AUTOSTART	AIX, Solaris-Betriebsumgebung	Standardwert: NO Werte: YES, NO
Gibt an, ob der DLFM-Server bei einem Warmstart des Betriebssystems immer automatisch gestartet wird. Diese Variable wird vom Script `dlfsmount` überprüft. Dieses Script wird während der Verarbeitung des Warmstarts aus der Datei `/etc/inittab` aufgerufen.		

Zugehörige Konzepte:

• „DB2-Registriervariablen und DB2-Umgebungsvariablen" auf Seite 565

Verschiedene Variablen

Tabelle 54. Verschiedene Variablen

Variablenname	Betriebssystem	Werte
Beschreibung		
DB2ADMINSERVER	Windows und UNIX	Standardwert=null
Gibt den DB2-Verwaltungsserver an.		
DB2CLIINIPATH	Alle	Standardwert=null
Dient zum Überschreiben des Standardpfads der DB2-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	UNIX	Standardwert=OFF Werte: OFF/NO/0 oder ON/YES/1
Auf UNIX-Plattformen vor Version 8 schrieb DB2 alle verbleibenden unvollständigen Transaktionen bei erfolgreicher Beendigung der Anweisung fest. In Version 8 wurde diese Funktionsweise geändert, so dass unvollständige Transaktionen bei Beendigung rückgängig gemacht werden. Diese Registriervariable gibt Benutzern mit Anwendungen, die von der früheren Funktionsweise abhängig sind, die Möglichkeit, diese Funktionsweise in Version 8 zu aktivieren. Beachten Sie, dass in Version 10 diese Registriervariable und die Funktionsweise mit Festschreibung bei Beendigung nicht weiter unterstützt werden. Benutzer sollten bestimmen, ob Anwendungen von ihnen, die vor Version 8 entwickelt wurden, weiterhin von dieser Funktionsweise abhängig sein sollten, und die entsprechenden expliziten COMMIT-Anweisungen nach Bedarf in die Anwendungen einfügen. Wenn die Registriervariable aktiviert ist, sollte sorgsam vermieden werden, neue Anwendungen zu implementieren, die keine expliziten COMMIT-Operationen vor ihrer Beendigung ausführen. Die meisten Benutzer sollten die Standardeinstellung dieser Registriervariablen belassen.		
DB2DEFPREP	Alle	Standardwert=NO Werte: ALL, YES oder NO
Simuliert das Laufzeitverhalten der Precompiler-Option DEFERRED_PREPARE für Anwendungen, die vor der Verfügbarkeit dieser Option vorkompiliert wurden. Wenn zum Beispiel eine Anwendung von DB2 Version 2.1.1 oder früher in einer Umgebung von DB2 Version 2.1.2 oder später ausgeführt würde, könnte der Parameter DB2DEFPREP verwendet werden, um das gewünschte Verhalten einer „verzögerten Vorbereitung" (Deferred Prepare) zu simulieren.		
DB2_DJ_COMM	Alle	Standardwert=null Werte: libdb2drda.a, libdb2net8.a, libdb2informix.a db2drda.dll, db2net8.dll, db2informix.a usw.
Gibt die Wrapper-Bibliotheken (Wrapper Libraries) an, die beim Start des Datenbankmanagers geladen werden. Durch Angeben dieser Variablen wird der Echtzeitaufwand für das Laden häufig benutzter Wrapper-Bibliotheken verringert. Andere Werte für andere Betriebssysteme werden unterstützt (die Erweiterung .dll ist für das Betriebssystem Windows NT, die Erweiterung .a für das Betriebssystem AIX). Die Bibliotheksnamen sind je nach Protokoll und Betriebssystem verschieden. Diese Variable wird ignoriert, sofern nicht der Parameter FEDERATED des Datenbankmanagers den Wert YES hat.		
DB2_DJ_INI	Alle	Standardwert: • UNIX: db2_exemplarverzeichnis/cfg/db2dj.ini • Windows: db2_installationsverzeichnis\cfg\db2dj.ini

Tabelle 54. Verschiedene Variablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		

Gibt den absoluten Pfadnamen der Konfigurationsdatei für den Zusammenschluss 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 DB2 Information Integrator bereitgestellten Wrappern verwendet.
 	
Das folgende Beispiel zeigt eine Konfigurationsdatei für den Zusammenschluss 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.		

DB2DMNBCKCTLR	Windows NT	Standardwert=null Werte: ? oder ein Domänenname
Wenn Sie den Namen der Domäne kennen, für die der DB2-Server der Sicherungsdomänencontroller ist, stellen Sie DB2DMNBCKCTLR=`DOMAIN_NAME` ein. Die Angabe für DOMÄNENNAME muss in Großbuchstaben erfolgen. Sie können DB2 die Domäne ermitteln lassen, für die die lokale Maschine ein Sicherungsdomänencontroller ist, indem Sie DB2DMNBCKCTLR=? einstellen. Wenn die Profilvariable DB2DMNBCKCTLR nicht oder auf Null eingestellt ist, führt DB2 die Identifikationsüberprüfung auf dem primären Domänencontroller aus. **Anmerkung:** DB2 verwendet standardmäßig keinen vorhandenen Sicherungsdomänencontroller, weil ein Sicherungsdomänencontroller die Synchronisation mit dem primären Domänencontroller verlieren und damit ein Sicherheitsproblem verursachen kann. Es kann zum Verlust der Synchronisation kommen, wenn die Sicherheitsdatenbank des primären Domänencontrollers aktualisiert wird, die Änderungen jedoch nicht an einen Sicherungsdomänencontroller weitergegeben werden. Der Grund hierfür könnten Netzwerklatenzzeiten oder ein nicht funktionierender Computersuchdienst sein.		
DB2_DOCHOST	Alle	Standardwert: http://publib.boulder.ibm.com/infocenter/db2help/ http://*hostname*, mit *hostname*= gültiger Hostname oder IP-Adresse
Gibt den Hostnamen an, auf dem 'DB2 Information - Unterstützung' installiert ist. Diese Variable kann bei der Installation von 'DB2 Information - Unterstützung' automatisch definiert werden, wenn die Option zur automatischen Konfiguration im DB2-Installationsassistenten ausgewählt wird.		
DB2_DOCPORT	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 von 'DB2 Information - Unterstützung' automatisch definiert werden, wenn die Option zur automatischen Konfiguration im DB2-Installationsassistenten ausgewählt wird.		
DB2_EXTSECURITY	Windows-Plattformen	Standardwert=ON Werte: ON oder OFF
Verhindert unbefugten Zugriff auf DB2 durch Sperren der DB2-Systemdateien. Zur Vermeidung potenzieller Probleme sollte diese Registriervariable nicht auf OFF gesetzt werden.		

Tabelle 54. Verschiedene Variablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
DB2_ENABLE_LDAP	Alle	Standardwert=NO Werte: YES oder NO
Gibt an, ob LDAP (Lightweight Directory Access Protocol) verwendet wird. LDAP ist eine Zugriffsmethode auf Verzeichnisservices.		
DB2_FALLBACK	Windows NT	Standardwert=OFF Werte: ON oder OFF
Diese Variable erlaubt Ihnen, alle Datenbankverbindungen bei der FALLBACK-Verarbeitung zu trennen. Sie wird in Verbindung mit der Unterstützung für die Funktionsübernahme in der Windows NT-Umgebung mit Microsoft Cluster Server (MSCS) verwendet. Wenn DB2_FALLBACK nicht oder auf OFF gesetzt ist und eine Datenbankverbindung bei der FALLBACK-Operation besteht, kann die DB2-Ressource nicht offline genommen werden. Das bedeutet, dass die FALLBACK-Verarbeitung fehlschlägt.		
DB2_FMP_COMM_HEAPSZ	Windows, alle UNIX-Systeme außer AIX	20 MB oder ausreichend Speicherbereich, um 10 abgeschirmte Routinen ausführen zu können (je nachdem, welcher Wert größer ist)
Diese Variable gibt (in Seiten zu 4 KB) die Größe des Pools an, der für den Aufruf abgeschirmter Routinen wie beispielsweise 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 entsprechenden Wert senken. Wird dieser Wert auf 0 gesetzt, wird keine Gruppe erstellt, so dass keine abgeschirmten Routinen aufgerufen werden können. Dies bedeutet außerdem, dass der Diagnosemonitor und die Funktionalität zur automatischen Datenbankpflege (z. B. automatische Sicherungen, Statistikerfassungen und Reorganisationen) inaktiviert wird, da diese Funktionalität von der Infrastruktur für abgeschirmte Routinen abhängig ist.		
DB2_GRP_LOOKUP	Windows NT	Standardwert=null Werte: LOCAL, DOMAIN
Mit dieser Variable wird DB2 mitgeteilt, wo Benutzerkonten zu überprüfen und Gruppenmitgliedsuchen durchzuführen sind. Setzen Sie diese Variable auf LOCAL, um DB2 zu veranlassen, das Aufzählen von Gruppen und die Überprüfung von Benutzerkonten immer auf dem DB2-Server durchzuführen. Setzen Sie die Variable auf DOMAIN, um DB2 zu veranlassen, das Aufzählen von Gruppen und die Überprüfung von Benutzerkonten immer in der Windows NT-Domäne durchzuführen, zu der das Benutzerkonto gehört.		
DB2_HADR_BUF_SIZE	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. Diese Variable sollte im Bereitschaftsexemplar 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.		
DB2LDAP_BASEDN	Alle	Standardwert=null Werte: ein beliebiger Basisdomänenname.

Tabelle 54. Verschiedene Variablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
Gibt den Basisdomänennamen für das LDAP-Verzeichnis an.		
DB2LDAPCACHE	Alle	Standardwert=YES Werte: YES oder NO

Gibt an, dass der LDAP-Cache aktiviert werden muss. 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 DB DIR`
`REFRESH LDAP NODE DIR`
 	
Diese Befehle aktualisieren das Datenbank- und das Knotenverzeichnis und entfernen daraus falsche Einträge.		

DB2LDAP_CLIENT_PROVIDER	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 Registriervariable wird dazu verwendet, den von DB2 zu verwendenden LDAP-Client explizit auszuwählen. **Anmerkung:** Verwenden Sie zum Anzeigen des aktuellen Werts dieser Registriervariablen den Befehl `db2set`: `db2set DB2LDAP_CLIENT_PROVIDER`		
DB2LDAPHOST	Alle	Standardwert=null Werte: ein beliebiger gültiger Hostname.
Gibt den Hostnamen des Standorts für das LDAP-Verzeichnis an.		
DB2LDAP_KEEP_CONNECTION	Alle	Standardwert=YES Werte: YES, NO
Gibt an, ob DB2 interne LDAP-Verbindungskennungen in den Cache stellt. Wenn diese Variable auf NO gesetzt wird, stellt DB2 die LDAP-Verbindungskennungen 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. Setzen Sie diese Variable standardmäßig auf den Wert YES, um die optimale Leistung zu erhalten. Die Registriervariable DB2LDAP_KEEP_CONNECTION ist nur als Profilregistriervariable der globalen Ebene in LDAP implementiert, so dass sie mit dem Befehl **db2set** unter Angabe der Option -gl wie folgt definiert werden muss: `db2set -gl DB2LDAP_KEEP_CONNECTION=NO`		
DB2LDAP_SEARCH_SCOPE	Alle	Standardwert = DOMAIN Werte: LOCAL, DOMAIN, GLOBAL
Gibt den Bereich für die Suche nach Informationen an, die in Partitionen oder Domänen im Lightweight Directory Access Protocol (LDAP) gefunden werden. Der Wert LOCAL inaktiviert das Suchen im LDAP-Verzeichnis. Bei dem Wert DOMAIN wird das LDAP nur nach der aktuellen Verzeichnispartition durchsucht. Bei dem Wert GLOBAL wird das LDAP in allen Verzeichnispartitionen durchsucht, bis das Objekt gefunden wird.		
DB2_LOAD_COPY_NO_OVERRIDE	Alle	Standardwert: NONRECOVERABLE Werte: COPY YES, NONRECOVERABLE

Tabelle 54. Verschiedene Variablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
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 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.		
DB2LOADREC	Alle	Standardwert=null
Dient zum Überschreiben der Speicherposition der Ladekopie bei einer aktualisierenden Wiederherstellung (ROLLFORWARD). Wenn der Benutzer die physische Speicherposition der Ladekopie geändert hat, muss die Variable DB2LOADREC vor dem Ausführen der aktualisierenden Wiederherstellung definiert werden.		
DB2LOCK_TO_RB	Alle	Standardwert=null Werte: STATEMENT
Definiert, ob durch Sperrenzeitlimits die gesamte Transaktion oder nur die aktuelle Anweisung rückgängig gemacht werden soll. Wenn DB2LOCK_TO_RB den Wert `STATEMENT` hat, wird aufgrund eines Zeitlimits für Sperren nur die aktuelle Anweisung rückgängig gemacht. Jeder andere Wert für diesen Parameter sorgt dafür, dass die gesamte Transaktion rückgängig gemacht wird.		
DB2_NEWLOGPATH2	UNIX	Standardwert=0 Werte: 0 oder 1
Dieser Parameter gibt Ihnen die Möglichkeit anzugeben, ob über einen zweiten Pfad eine doppelte Protokollierung eingerichtet werden soll. Der sekundäre Pfadname wird generiert, indem eine „2" an den aktuellen Wert des Datenbankkonfigurationsparameters *logpath* angefügt wird.		
DB2NOEXITLIST	Alle	Standardwert=OFF Werte: ON oder OFF
Wenn diese Variable definiert ist, weist sie DB2 an, keine Ausgangslistenroutine in Anwendungen zu installieren und keine COMMIT-Operation auszuführen. Normalerweise installiert DB2 eine Prozessausgangslistenroutine in Anwendungen, und die Ausgangslistenroutine führt eine COMMIT-Operation aus, wenn die Anwendung normal beendet wird. Bei Anwendungen, die die DB2-Bibliothek dynamisch laden und aus dem Speicher entfernen, bevor die Anwendung beendet ist, schlägt der Aufruf der Ausgangslistenroutine fehl, weil die Routine nicht mehr in der Anwendung geladen ist. Wenn Ihre Anwendung so arbeitet, müssen Sie die Variable DB2NOEXITLIST angeben und sicherstellen, dass Ihre Anwendung alle erforderlichen COMMIT-Operationen explizit aufruft.		
DB2OLDEVMON	Alle	Werte: Ereignismonitornamen durch Kommata getrennt*ergmon1*, *ergmon2*, ...
Gibt die Namen von Ereignismonitoren an, die Daten in einem Format vor Version 6 schreiben. In DB2 Version 6 wurde der selbstbeschreibende Datenstrom zum Standardformat der Systemmonitorausgabe in Dateien und Pipes. Vor Version 6 wurden Systemmonitordaten in festen Datenstrukturen zurückgegeben.		
DB2REMOTEPREG	Windows NT	Standardwert=null Werte: ein beliebiger gültiger Windows NT-Maschinenname
Definiert den Namen der fernen Maschine, die die Win32-Registrierdatenbankliste von DB2-Exemplarprofilen und DB2-Exemplaren enthält. Der Wert für DB2REMOTEPREG sollte nur einmal nach der Installation von DB2 definiert und anschließend nicht mehr geändert werden. Verwenden Sie diese Variable mit großer Vorsicht.		
DB2ROUTINE_DEBUG	AIX und Windows NT	Standardwert=OFF Werte: ON, OFF

Tabelle 54. Verschiedene Variablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
Gibt an, ob die Fehlerbehebungsfunktion (Debug) für gespeicherte Java-Prozeduren aktiviert wird. Wenn sie nicht gerade Fehler in gespeicherten Java-Prozeduren beheben, sollten Sie den Standardwert OFF verwenden. Die Aktivierung des Fehlerbehebungsmodus hat Auswirkungen auf die Leistung.		
DB2SATELLITEID	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.		
DB2SORT	Alle, nur Server	Standardwert=null
Definiert die Speicherposition einer während der Laufzeit durch das Dienstprogramm LOAD zu ladenden Bibliothek. 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.		
DB2SYSTEM	Windows und UNIX	Standardwert=null
Definiert den Namen, der von Ihren Benutzern und Datenbankadministratoren zur Identifizierung des DB2-Serversystems verwendet wird. Dieser Name sollte nach Möglichkeit innerhalb Ihres Netzwerks eindeutig sein. Dieser Name wird auf der Systemebene der Objektbaumstruktur in der Steuerzentrale angezeigt, um Administratoren bei der Identifizierung von Server-Systemen zu helfen, die von der Steuerzentrale aus verwaltet werden können. Bei Verwendung der Funktion zum Durchsuchen des Netzwerks der „Clientkonfiguration - Unterstützung" gibt DB2-Discovery diesen Namen zurück und er wird auf der Systemebene der sich ergebenden Baumstruktur angezeigt. 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 NT setzt das SETUP-Programm 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_VENDOR_INI	AIX, HP-UX, Solaris-Betriebsumgebung und Windows	Standardwert=null Werte: eine beliebige gültige Angabe für Pfad und Datei
Verweist auf eine Datei, die alle herstellerspezifischen Umgebungseinstellungen enthält. Der Wert wird beim Start des Datenbankmanagers gelesen.		
DB2_XBSA_LIBRARY	AIX, HP-UX, Solaris-Betriebsumgebung und Windows	Standardwert=null Werte: eine beliebige gültige Angabe für Pfad und Datei

Verweist auf die werkseitige XBSA-Bibliothek. Unter AIX muss die Einstellung das gemeinsam benutzte Objekt enthalten, sofern es nicht shr.o heißt. Für HP-UX, die Solaris-Betriebsumgebung und Windows NT wird der Name des gemeinsam benutzten Objekts nicht benötigt. Wenn z. B. das NetWorker Business Suite Module für DB2 von Legato verwendet werden soll, muss die Registriervariable 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. Zum Beispiel:
`db2 backup db sample use XBSA`
`db2 restore db sample use XBSA`		

Zugehörige Konzepte:

• „DB2-Registriervariablen und DB2-Umgebungsvariablen" auf Seite 565

Anhang B. EXPLAIN-Tabellen

EXPLAIN-Tabellen

Die EXPLAIN-Tabellen erfassen Zugriffspläne, wenn das Programm EXPLAIN aktiviert ist. Die EXPLAIN-Tabellen müssen erstellt werden, bevor EXPLAIN aufgerufen werden kann. Sie können sie mit Hilfe der dokumentierten Tabellendefinitionen oder durch Aufrufen der Beispielprozedur für den Befehlszeilenprozessor erstellen, die in der Datei EXPLAIN.DDL bereitgestellt wird, die sich wiederum im Unterverzeichnis 'misc' des Verzeichnisses 'sqllib' befindet. Zum Aufrufen der Prozedur stellen Sie eine Verbindung zu der Datenbank her, in der die EXPLAIN-Tabellen benötigt werden, und geben den folgenden Befehl ein:

```
db2 -tf EXPLAIN.DDL
```

Das Füllen der EXPLAIN-Tabellen mit Werten durch die EXPLAIN-Einrichtung aktiviert weder Auslöser noch referenzielle Integritätsbedingungen oder Prüfungen auf Integritätsbedingungen. Wenn zum Beispiel ein Einfügeauslöser für die Tabelle EXPLAIN_INSTANCE definiert wäre und EXPLAIN-Informationen über eine auswählbare Anweisung erfasst würden, würde der Auslöser nicht aktiviert.

Zugehörige Referenzen:

- „Die Tabelle EXPLAIN_ARGUMENT" auf Seite 608
- „Die Tabelle EXPLAIN_OBJECT" auf Seite 615
- „Die Tabelle EXPLAIN_OPERATOR" auf Seite 618
- „Die Tabelle EXPLAIN_PREDICATE" auf Seite 620
- „Die Tabelle EXPLAIN_STREAM" auf Seite 625
- „Die Tabelle ADVISE_INDEX" auf Seite 627
- „Die Tabelle ADVISE_WORKLOAD" auf Seite 636
- „Die Tabelle EXPLAIN_INSTANCE" auf Seite 612
- „Die Tabelle EXPLAIN_STATEMENT" auf Seite 622
- „Die Tabelle ADVISE_INSTANCE" auf Seite 630
- „Die Tabelle ADVISE_MQT" auf Seite 631
- „Die Tabelle ADVISE_PARTITION" auf Seite 633
- „Die Tabelle ADVISE_TABLE" auf Seite 635

Die Tabelle EXPLAIN_ARGUMENT

Die Tabelle EXPLAIN_ARGUMENT enthält gegebenenfalls vorhandene, eindeutige Kenndaten für jeden einzelnen Operator.

Tabelle 55. Die Tabelle EXPLAIN_ARGUMENT. PK bedeutet, dass die Spalte Teil eines Primärschlüssels ist. FK bedeutet, dass die Spalte Teil eines Fremdschlüssels ist.

Spaltenname	Datentyp	Nullwert?	Schlüssel?	Beschreibung
EXPLAIN_REQUESTER	VARCHAR(128)	Nein	FK	Berechtigungs-ID des Initiators dieser EXPLAIN-Anforderung
EXPLAIN_TIME	TIMESTAMP	Nein	FK	Initialisierungszeitpunkt für die EXPLAIN-Anforderung
SOURCE_NAME	VARCHAR(128)	Nein	FK	Name des Pakets, das ausgeführt wurde, als die dynamische Anweisung mit EXPLAIN bearbeitet wurde, oder der Name der Quellendatei, als das statische SQL mit EXPLAIN bearbeitet wurde.
SOURCE_SCHEMA	VARCHAR(128)	Nein	FK	Schema oder Qualifikationsmerkmal der Quelle der EXPLAIN-Anforderung
SOURCE_VERSION	VARCHAR(64)	Nein	FK	Version der Quelle der EXPLAIN-Anforderung
EXPLAIN_LEVEL	CHAR(1)	Nein	FK	Ebene der EXPLAIN-Informationen, für die diese Zeile relevant ist
STMTNO	INTEGER	Nein	FK	Anweisungsnummer im Paket, auf die sich diese EXPLAIN-Informationen beziehen
SECTNO	INTEGER	Nein	FK	Abschnittsnummer im Paket, auf die sich diese EXPLAIN-Informationen beziehen
OPERATOR_ID	INTEGER	Nein	Nein	Eindeutige ID für diesen Operator in dieser Abfrage
ARGUMENT_TYPE	CHAR(8)	Nein	Nein	Der Argumenttyp für diesen Operator
ARGUMENT_VALUE	VARCHAR(1024)	Ja	Nein	Der Argumentwert für diesen Operator. NULL, wenn sich der Wert in LONG_ARGUMENT_VALUE befindet.
LONG_ARGUMENT_VALUE	CLOB(2M)	Ja	Nein	Der Argumentwert für diesen Operator, wenn der Text nicht in ARGUMENT_VALUE passt. NULL, wenn sich der Wert in ARGUMENT_VALUE befindet.

Tabelle 56. Werte der Spalten ARGUMENT_TYPE und ARGUMENT_VALUE

Wert für ARGUMENT_TYPE	Gültige Werte für ARGUMENT_VALUE	Beschreibung
AGGMODE	COMPLETE PARTIAL INTERMEDIATE FINAL	Partielle Bezugswerte für Spaltenberechnung
BITFLTR	TRUE FALSE	Die Hashverknüpfung verwendet einen Bitfilter zur Leistungsverbesserung.
BLD_LEVEL	DB2-Build-Kennung	Interne ID-Zeichenfolge für die Quellcodeversion
BLKLOCK	EXCLUSIVE INTENT EXCLUSIVE INTENT SHARE NONE SHARE UPDATE	Vorgesehene Sperre auf Blockebene
CSERQY	TRUE FALSE	Ferne Abfrage ist ein allgemeiner Unterausdruck (common subexpression).
CSETEMP	TRUE FALSE	Markierung für temporäre Tabelle über allgemeinen Unterausdruck
DIRECT	TRUE	Anzeiger für direkten Abruf

Tabelle 56. Werte der Spalten ARGUMENT_TYPE und ARGUMENT_VALUE (Forts.)

Wert für ARGUMENT_TYPE	Gültige Werte für ARGUMENT_VALUE	Beschreibung
DSTSEVER	Servername	Zielserver (Absendeserver)
DUPLWARN	TRUE FALSE	Markierung für Warnung, dass gleiche Werte auftraten
EARLYOUT	LEFT RIGHT NONE	Anzeiger für frühes Verlassen (Early out)
ENVVAR	Jede Zeile dieses Typs enthält die folgenden Angaben: • Name der Umgebungsvariable • Wert der Umgebungsvariable	Sich auf das Optimierungsprogramm auswirkende Umgebungsvariable
FETCHMAX	IGNORE INTEGER	Außerkraftsetzen des Werts für das Argument MAXPAGES im Operator FETCH
GREEDY	TRUE	Gibt an, dass das Optimierungsprogramm den GREEDY-Algorithmus zur Zugriffsplanung verwendete.
GROUPBYC	TRUE FALSE	Gibt an, ob Spalten mit der Klausel Group By angegeben wurden.
GROUPBYN	Ganze Zahl	Anzahl der Vergleichsspalten
GROUPBYR	Jede Zeile dieses Typs enthält die folgenden Angaben: • Ordinalzahl der Spalte in der Klausel Group By (gefolgt von einem Doppelpunkt und einem Leerzeichen) • Name der Spalte	Anforderung der Klausel Group By
INNERCOL	Jede Zeile dieses Typs enthält die folgenden Angaben: • Ordinalzahl der Spalte in der Reihenfolge (gefolgt von einem Doppelpunkt und einem Leerzeichen) • Name der Spalte • Wert für Reihenfolge **(A)** Aufsteigend **(D)** Absteigend	Innere Ordnungsspalten
ISCANMAX	IGNORE INTEGER	Außerkraftsetzen des Werts für das Argument MAXPAGES im Operator ISCAN
JN_INPUT	INNER OUTER	Gibt an, ob Operator der Operator ist, der INNER oder OUTER für eine JOIN-Operation bereitstellt.
LISTENER	TRUE FALSE	Anzeiger für empfangsbereite Tabellenwarteschlange (Listener)
MAXPAGES	ALL NONE INTEGER	Maximale Anzahl der erwarteten Seiten für Vorablesezugriff
MAXRIDS	NONE INTEGER	Die maximale Anzahl der Zeilen-IDs muss in jeder Anforderung für Vorablesezugriff über Listen enthalten sein.
NUMROWS	INTEGER	Anzahl der erwarteten, zu sortierenden Zeilen
ONEFETCH	TRUE FALSE	Anzeiger für eine Abrufoperation (FETCH)

Tabelle 56. Werte der Spalten ARGUMENT_TYPE und ARGUMENT_VALUE (Forts.)

Wert für ARGUMENT_TYPE	Gültige Werte für ARGUMENT_VALUE	Beschreibung
OUTERCOL	Jede Zeile dieses Typs enthält die folgenden Angaben: • Ordinalzahl der Spalte in der Reihenfolge (gefolgt von einem Doppelpunkt und einem Leerzeichen) • Name der Spalte • Wert für Reihenfolge **(A)** Aufsteigend **(D)** Absteigend	Äußere Ordnungsspalten
OUTERJN	LEFT RIGHT FULL LEFT (ANTI) RIGHT (ANTI)	Anzeiger für äußere Verknüpfung
PARTCOLS	Name der Spalte	Partitionierungsspalten für den Operator
PREFETCH	LIST NONE SEQUENTIAL	Art des verfügbaren Vorablesezugriffs
RMTQTEXT	Abfragetext	Ferner Abfragetext
RNG_PROD	Funktionsname	Wertebereich erzeugende Funktion für erweiterten Indexzugriff
ROWLOCK	EXCLUSIVE NONE REUSE SHARE SHORT (INSTANT) SHARE UPDATE	Vorgesehene Zeilensperre (Row Lock Intent)
ROWWIDTH	INTEGER	Länge der zu sortierenden Zeile
RSUFFIX	Abfragetext	Suffix für fernes SQL
SCANDIR	FORWARD REVERSE	Suchrichtung
SCANGRAN	INTEGER	Partitionsinterne Parallelität, Granularität der partitionsinternen Parallelsuche, ausgedrückt in SCANUNITs
SCANTYPE	LOCAL PARALLEL	Partitionsinterne Parallelität, Index- oder Tabellensuche
SCANUNIT	ROW PAGE	Partitionsinterne Parallelität, Einheit für Suchgranularität
SHARED	TRUE	Partitionsinterne Parallelität, Anzeiger für gemeinsam benutztes TEMP
SLOWMAT	TRUE FALSE	Markierung für langsame Materialisierung (Slow Materialization)
SNGLPROD	TRUE FALSE	Sortierung durch partitionsinterne Parallelität oder Tabelle TEMP von einzelnem Agenten erstellt.

Tabelle 56. Werte der Spalten ARGUMENT_TYPE und ARGUMENT_VALUE (Forts.)

Wert für ARGUMENT-_TYPE	Gültige Werte für ARGUMENT_VALUE	Beschreibung
SORTKEY	Jede Zeile dieses Typs enthält die folgenden Angaben: • Ordinalzahl der Spalte im Schlüssel (gefolgt von einem Doppelpunkt und einem Leerzeichen) • Name der Spalte • Wert für Reihenfolge **(A)** Aufsteigend **(D)** Absteigend	Sortierschlüsselspalten
SORTTYPE	PARTITIONED SHARED ROUND ROBIN REPLICATED	Partitionsinterne Parallelität, Sortiertyp
SRCSEVER	Servername	Quellenserver (Empfangsserver)
STREAM	TRUE FALSE	Ferne Quelle im Datenstrommodus (STREAMING)
TABLOCK	EXCLUSIVE INTENT EXCLUSIVE INTENT NONE INTENT SHARE REUSE SHARE SHARE INTENT EXCLUSIVE SUPER EXCLUSIVE UPDATE	Vorgesehene Tabellensperre (Table Lock Intent)
TEMPSIZE	INTEGER	Seitengröße der temporären Tabelle
TQDEGREE	INTEGER	Partitionsinterne Parallelität, Anzahl von Subagenten, die auf die Tabellenwarteschlange zugreifen
TQMERGE	TRUE FALSE	Angabe für (sortiertes) Mischen von Tabellenwarteschlangen
TQREAD	READ AHEAD STEPPING SUBQUERY STEPPING	Lesemerkmal für Tabellenwarteschlange
TQSEND	BROADCAST DIRECTED SCATTER SUBQUERY DIRECTED	Sendemerkmal für Tabellenwarteschlange
TQTYPE	LOCAL	Partitionsinterne Parallelität, Tabellenwarteschlange
TRUNCSRT	TRUE	Einschränkendes Sortieren (beschränkt die Anzahl erzeugter Zeilen)
UNIQUE	TRUE FALSE	Anzeiger für Eindeutigkeit
UNIQKEY	Jede Zeile dieses Typs enthält die folgenden Angaben: • Ordinalzahl der Spalte im Schlüssel (gefolgt von einem Doppelpunkt und einem Leerzeichen) • Name der Spalte	Eindeutige Schlüsselspalten
VOLATILE	TRUE	Flüchtige Tabelle

Die Tabelle EXPLAIN_INSTANCE

Die Tabelle EXPLAIN_INSTANCE ist die Hauptsteuertabelle für alle EXPLAIN-Informationen. Jede Datenzeile in den EXPLAIN-Tabellen ist explizit mit einer eindeutigen Zeile in dieser Tabelle verbunden. Die Tabelle EXPLAIN_INSTANCE enthält grundlegende Informationen über die Quelle der SQL-Anweisungen, für die EXPLAIN-Informationen ausgegeben werden, sowie Informationen über die Umgebung, in der die Ausgabe von EXPLAIN-Informationen stattfand.

Tabelle 57. Tabelle EXPLAIN_INSTANCE. PK bedeutet, dass die Spalte Teil eines Primärschlüssels ist. FK bedeutet, dass die Spalte Teil eines Fremdschlüssels ist.

Spaltenname	Datentyp	Nullwert?	Schlüssel?	Beschreibung
EXPLAIN_REQUESTER	VARCHAR(128)	Nein	PK	Berechtigungs-ID des Initiators dieser EXPLAIN-Anforderung
EXPLAIN_TIME	TIMESTAMP	Nein	PK	Initialisierungszeitpunkt für die EXPLAIN-Anforderung
SOURCE_NAME	VARCHAR(128)	Nein	PK	Name des Pakets, das ausgeführt wurde, als die dynamische Anweisung mit EXPLAIN bearbeitet wurde, oder der Name der Quellendatei, als das statische SQL mit EXPLAIN bearbeitet wurde.
SOURCE_SCHEMA	VARCHAR(128)	Nein	PK	Schema oder Qualifikationsmerkmal der Quelle der EXPLAIN-Anforderung
SOURCE_VERSION	VARCHAR(64)	Nein	PK	Version der Quelle der EXPLAIN-Anforderung
EXPLAIN_OPTION	CHAR(1)	Nein	Nein	Gibt an, welche EXPLAIN-Informationen bei dieser Anforderung angefordert wurden. Die folgenden Werte sind möglich: **P** PLANAUSWAHL
SNAPSHOT_TAKEN	CHAR(1)	Nein	Nein	Gibt an, ob eine EXPLAIN-Momentaufnahme für diese Anforderung erstellt wurde. Die folgenden Werte sind möglich: **Y** Es wurde eine oder mehrere EXPLAIN-Momentaufnahmen erstellt und in der Tabelle EXPLAIN_STATEMENT gespeichert. Darüber hinaus wurden auch reguläre EXPLAIN-Informationen erfasst. **N** Es wurde keine EXPLAIN-Momentaufnahme erstellt. Die regulären EXPLAIN-Informationen wurden jedoch erfasst. **O** Es wurde nur eine EXPLAIN-Momentaufnahme erstellt. Die regulären EXPLAIN-Informationen wurden nicht erfasst.
DB2_VERSION	CHAR(7)	Nein	Nein	Release-Nummer des Produkts DB2 Universal Database, das diese EXPLAIN-Anforderung verarbeitete. Das Format ist vv.rr.m, wobei Folgendes gilt: **vv** Versionsnummer **rr** Release-Nummer **m** Nummer des Wartungs-Release
SQL_TYPE	CHAR(1)	Nein	Nein	Gibt an, ob das EXPLAIN-Exemplar für statisches oder dynamisches SQL galt. Die folgenden Werte sind möglich: **S** Statisches SQL **D** Dynamisches SQL
QUERYOPT	INTEGER	Nein	Nein	Gibt die vom SQL-Compiler zum Zeitpunkt des EXPLAIN-Aufrufs verwendete Optimierungsklasse an. Der Wert gibt an, welche Stufe der Abfrageoptimierung durch den SQL-Compiler für die mit EXPLAIN bearbeiteten SQL-Anweisungen ausgeführt wurde.

Tabelle 57. Tabelle EXPLAIN_INSTANCE (Forts.). PK bedeutet, dass die Spalte Teil eines Primärschlüssels ist. FK bedeutet, dass die Spalte Teil eines Fremdschlüssels ist.

Spaltenname	Datentyp	Nullwert?	Schlüssel?	Beschreibung
BLOCK	CHAR(1)	Nein	Nein	Gibt an, welche Art von Cursorblockung beim Kompilieren der SQL-Anweisungen ausgeführt wurde. Weitere Informationen finden Sie in der Spalte BLOCK in der Katalogsicht SYSCAT.PACKAGES. Die folgenden Werte sind möglich: **N** Keine Blockung **U** Blockung eindeutiger Cursor **B** Blockung aller Cursor
ISOLATION	CHAR(2)	Nein	Nein	Gibt an, welche Isolationsstufe beim Kompilieren der SQL-Anweisungen verwendet wurde. Weitere Informationen finden Sie in der Spalte ISOLATION in der Katalogsicht SYSCAT.PACKAGES. Die folgenden Werte sind möglich: **RR** Wiederholtes Lesen (Repeatable Read) **RS** Lesestabilität (Read Stability) **CS** Cursorstabilität (Cursor Stability) **UR** Nicht festgeschriebener Lesevorgang (Uncommitted Read)
BUFFPAGE	INTEGER	Nein	Nein	Enthält den Wert des Konfigurationsparameters BUFFPAGE für die Datenbank zum Zeitpunkt des Aufrufs von EXPLAIN.
AVG_APPLS	INTEGER	Nein	Nein	Enthält den Wert des Konfigurationsparameters AVG_APPLS zum Zeitpunkt des Aufrufs von EXPLAIN.
SORTHEAP	INTEGER	Nein	Nein	Enthält den Wert des Konfigurationsparameters SORTHEAP für die Datenbank zum Zeitpunkt des Aufrufs von EXPLAIN.
LOCKLIST	INTEGER	Nein	Nein	Enthält den Wert des Konfigurationsparameters LOCKLIST für die Datenbank zum Zeitpunkt des Aufrufs von EXPLAIN.
MAXLOCKS	SMALLINT	Nein	Nein	Enthält den Wert des Konfigurationsparameters MAXLOCKS für die Datenbank zum Zeitpunkt des Aufrufs von EXPLAIN.
LOCKS_AVAIL	INTEGER	Nein	Nein	Enthält die Anzahl der Sperren, die nach Annahme des Optimierungsprogramms für jeden Benutzer verfügbar sind. (Von den Konfigurationsparametern LOCKLIST und MAXLOCKS abgeleitet.)
CPU_SPEED	DOUBLE	Nein	Nein	Enthält den Wert des Konfigurationsparameters CPUSPEED für den Datenbankmanager zum Zeitpunkt des Aufrufs von EXPLAIN.
REMARKS	VARCHAR(254)	Ja	Nein	Vom Benutzer angegebener Kommentar
DBHEAP	INTEGER	Nein	Nein	Enthält den Wert des Konfigurationsparameters DBHEAP für die Datenbank zum Zeitpunkt des Aufrufs von EXPLAIN.
COMM_SPEED	DOUBLE	Nein	Nein	Enthält den Wert des Konfigurationsparameters COMM_BANDWIDTH für die Datenbank zum Zeitpunkt des Aufrufs von EXPLAIN.
PARALLELISM	CHAR(2)	Nein	Nein	Die folgenden Werte sind möglich: • N=Keine Parallelität • P=Partitionsinterne Parallelität • IP=Partitionsübergreifende Parallelität • BP=Partitionsinterne Parallelität und partitionsübergreifende Parallelität

Tabelle 57. Tabelle EXPLAIN_INSTANCE (Forts.). PK bedeutet, dass die Spalte Teil eines Primärschlüssels ist. FK bedeutet, dass die Spalte Teil eines Fremdschlüssels ist.

Spaltenname	Datentyp	Nullwert?	Schlüssel?	Beschreibung
DATAJOINER	CHAR(1)	Nein	Nein	Die folgenden Werte sind möglich: • N=Zugriffsplan von Systemen mit nicht zusammengeschlossenen Datenbanken • Y=Zugriffsplan von Systemen mit zusammengeschlossenen Datenbanken

Die Tabelle EXPLAIN_OBJECT

Die Tabelle EXPLAIN_OBJECT identifiziert die Datenobjekte, die für den Zugriffsplan erforderlich sind, der zur Ausführung der SQL-Anweisung generiert wurde.

Tabelle 58. Tabelle EXPLAIN_OBJECT. PK bedeutet, dass die Spalte Teil eines Primärschlüssels ist. FK bedeutet, dass die Spalte Teil eines Fremdschlüssels ist.

Spaltenname	Datentyp	Nullwert?	Schlüssel?	Beschreibung
EXPLAIN_REQUESTER	VARCHAR(128)	Nein	FK	Berechtigungs-ID des Initiators dieser EXPLAIN-Anforderung
EXPLAIN_TIME	TIMESTAMP	Nein	FK	Initialisierungszeitpunkt für die EXPLAIN-Anforderung
SOURCE_NAME	VARCHAR(128)	Nein	FK	Name des Pakets, das ausgeführt wurde, als die dynamische Anweisung mit EXPLAIN bearbeitet wurde, oder der Name der Quellendatei, als das statische SQL mit EXPLAIN bearbeitet wurde.
SOURCE_SCHEMA	VARCHAR(128)	Nein	FK	Schema oder Qualifikationsmerkmal der Quelle der EXPLAIN-Anforderung
SOURCE_VERSION	VARCHAR(64)	Nein	FK	Version der Quelle der EXPLAIN-Anforderung
EXPLAIN_LEVEL	CHAR(1)	Nein	FK	Ebene der EXPLAIN-Informationen, für die diese Zeile relevant ist
STMTNO	INTEGER	Nein	FK	Anweisungsnummer im Paket, auf die sich diese EXPLAIN-Informationen beziehen.
SECTNO	INTEGER	Nein	FK	Abschnittsnummer im Paket, auf die sich diese EXPLAIN-Informationen beziehen
OBJECT_SCHEMA	VARCHAR(128)	Nein	Nein	Schema, zu dem dieses Objekt gehört
OBJECT_NAME	VARCHAR(128)	Nein	Nein	Name des Objekts
OBJECT_TYPE	CHAR(2)	Nein	Nein	Beschreibende Kennung für den Objekttyp.
CREATE_TIME	TIMESTAMP	Ja	Nein	Zeitpunkt der Erstellung des Objekts. Null, wenn es sich um eine Tabellenfunktion handelt.
STATISTICS_TIME	TIMESTAMP	Ja	Nein	Zeitpunkt der letzten Statistikaktualisierung für dieses Objekt. Null, wenn keine Statistik für dieses Objekt vorhanden ist.
COLUMN_COUNT	SMALLINT	Nein	Nein	Anzahl der Spalten in diesem Objekt
ROW_COUNT	INTEGER	Nein	Nein	Geschätzte Anzahl Zeilen in diesem Objekt.
WIDTH	INTEGER	Nein	Nein	Durchschnittliche Länge des Objekts in Byte. Bei einem Index auf -1 gesetzt.
PAGES	INTEGER	Nein	Nein	Geschätzte Anzahl Seiten, die das Objekt im Pufferpool einnimmt. Für eine Tabellenfunktion auf -1 gesetzt.
DISTINCT	CHAR(1)	Nein	Nein	Gibt an, ob die Zeilen in dem Objekt eindeutig sind (d. h. keine doppelten Werte enthalten). Mögliche Werte: **Y** Ja **N** Nein
TABLESPACE_NAME	VARCHAR(128)	Ja	Nein	Name des Tabellenbereichs, in dem dieses Objekt gespeichert ist. Null, wenn kein Tabellenbereich benutzt wird.
OVERHEAD	DOUBLE	Nein	Nein	Geschätzter Gesamtaufwand in Millisekunden für eine einzelne wahlfreie Ein-/Ausgabeoperation mit dem angegebenen Tabellenbereich. Berücksichtigt die Aufwände für die Steuereinheit, für Plattensuchen und die Latenzzeit. Wert -1, wenn kein Tabellenbereich benutzt wird.

Tabelle 58. Tabelle EXPLAIN_OBJECT (Forts.). PK bedeutet, dass die Spalte Teil eines Primärschlüssels ist. FK bedeutet, dass die Spalte Teil eines Fremdschlüssels ist.

Spaltenname	Datentyp	Nullwert?	Schlüssel?	Beschreibung
TRANSFER_RATE	DOUBLE	Nein	Nein	Geschätzte Zeit (in Millisekunden) zum Lesen einer Datenseite vom angegebenen Tabellenbereich. Wert -1, wenn kein Tabellenbereich benutzt wird.
PREFETCHSIZE	INTEGER	Nein	Nein	Die Anzahl der Datenseiten, die gelesen werden, wenn der Vorablesezugriff (Prefetch) aktiv ist. Für eine Tabellenfunktion auf -1 gesetzt.
EXTENTSIZE	INTEGER	Nein	Nein	Größe des Speicherbereichs in Datenseiten. Dies ist die Anzahl der 4-KB-Seiten, die in einen Behälter des Tabellenbereichs geschrieben werden, bevor zum nächsten Behälter gewechselt wird. Für eine Tabellenfunktion auf -1 gesetzt.
CLUSTER	DOUBLE	Nein	Nein	Grad der Datenclusterbildung bezüglich des Index. Wenn >= 1, ist dies der Wert CLUSTERRATIO. Wenn >= 0 und < 1, ist dies der Wert CLUSTERFACTOR. Für eine Tabelle, Tabellenfunktion oder wenn diese Statistik nicht verfügbar ist, auf den Wert -1 gesetzt.
NLEAF	INTEGER	Nein	Nein	Die Anzahl der Blattseiten (Leaf pages), die von den Werten dieses Indexobjekts eingenommen werden. Für eine Tabelle, Tabellenfunktion oder wenn diese Statistik nicht verfügbar ist, auf den Wert -1 gesetzt.
NLEVELS	INTEGER	Nein	Nein	Die Anzahl von Indexstufen in der Baumstruktur dieses Indexobjekts. Für eine Tabelle, Tabellenfunktion oder wenn diese Statistik nicht verfügbar ist, auf den Wert -1 gesetzt.
FULLKEYCARD	BIGINT	Nein	Nein	Anzahl der unterschiedlichen vollständigen Schlüsselwerte in diesem Indexobjekt. Für eine Tabelle, Tabellenfunktion oder wenn diese Statistik nicht verfügbar ist, auf den Wert -1 gesetzt.
OVERFLOW	INTEGER	Nein	Nein	Gesamtzahl der Überlaufsätze in der Tabelle. Für einen Index, eine Tabellenfunktion oder wenn diese Statistik nicht verfügbar ist, auf den Wert -1 gesetzt.
FIRSTKEYCARD	BIGINT	Nein	Nein	Anzahl der eindeutigen ersten Schlüsselwerte. Für eine Tabelle, Tabellenfunktion oder wenn diese Statistik nicht verfügbar ist, auf den Wert –1 gesetzt.
FIRST2KEYCARD	BIGINT	Nein	Nein	Anzahl der eindeutigen ersten Schlüsselwerte bei Verwendung der ersten {2,3,4} Spalten des Index. Für eine Tabelle, Tabellenfunktion oder wenn diese Statistik nicht verfügbar ist, auf den Wert –1 gesetzt.
FIRST3KEYCARD	BIGINT	Nein	Nein	
FIRST4KEYCARD	BIGINT	Nein	Nein	
SEQUENTIAL_PAGES	INTEGER	Nein	Nein	Anzahl der Blattseiten (Leaf pages), die sich mit wenigen oder keinen größeren Abständen nach Indexschlüsseln sortiert auf der Platte befinden. Für eine Tabelle, Tabellenfunktion oder wenn diese Statistik nicht verfügbar ist, auf den Wert –1 gesetzt.
DENSITY	INTEGER	Nein	Nein	Verhältnis der SEQUENTIAL_PAGES zur Anzahl der Seiten im durch den Index belegten Seitenbereich; ausgedrückt als Prozentwert (ganze Zahl zwischen 0 und 100). Für eine Tabelle, Tabellenfunktion oder wenn diese Statistik nicht verfügbar ist, auf den Wert –1 gesetzt.
STATS_SRC	CHAR(1)	Nein	Nein	Gibt die Quelle für die Statistiken an. Auf den Wert 1 gesetzt, wenn von einem einzigen Knoten.
AVERAGE_SEQUENCE_GAP	DOUBLE	Nein	Nein	Lücke zwischen den Seitenfolgen
AVERAGE_SEQUENCE_FETCH_GAP	DOUBLE	Nein	Nein	Lücke zwischen Seitenfolgen beim Abrufen des Index
AVERAGE_SEQUENCE_PAGES	DOUBLE	Nein	Nein	Durchschnittliche Anzahl von Indexseiten, auf die in Reihenfolge zugegriffen werden kann

Tabelle 58. Tabelle EXPLAIN_OBJECT (Forts.). PK bedeutet, dass die Spalte Teil eines Primärschlüssels ist. FK bedeutet, dass die Spalte Teil eines Fremdschlüssels ist.

Spaltenname	Datentyp	Nullwert?	Schlüssel?	Beschreibung
AVERAGE_SEQUENCE_FETCH_PAGES	DOUBLE	Nein	Nein	Durchschnittliche Anzahl von Tabellenseiten, auf die in Reihenfolge zugegriffen werden kann
AVERAGE_RANDOM_PAGES	DOUBLE	Nein	Nein	Durchschnittliche Anzahl von Indexseiten, auf die zwischen den sequenziellen Seitenzugriffen wahlfrei zugegriffen werden muss
AVERAGE_RANDOM_FETCH_PAGES	DOUBLE	Nein	Nein	Durchschnittliche Anzahl von Tabellenseiten, auf die ein wahlfreier Zugriff erfolgt, zwischen sequenziellen Seitenzugriffen beim Abrufen über den Index
NUMRIDS	BIGINT	Nein	Nein	Gesamtanzahl von Satz-IDs (RIDs) im Index
NUMRIDS_DELETED	BIGINT	Nein	Nein	Gesamtanzahl quasi-gelöschter Satz-IDs (RIDs) im Index
NUM_EMPTY_LEAFS	BIGINT	Nein	Nein	Gesamtanzahl leerer Blattseiten im Index
ACTIVE_BLOCKS	BIGINT	Nein	Nein	Gesamtanzahl aktiver MDC-Blöcke in der Tabelle

Tabelle 59. Mögliche Werte für OBJECT_TYPE

Wert	Beschreibung
IX	Index
TA	Tabelle
TF	Tabellenfunktion

Die Tabelle EXPLAIN_OPERATOR

Die Tabelle EXPLAIN_OPERATOR enthält alle Operatoren, die der SQL-Compiler zum Ausführen der SQL-Anweisung benötigt.

Tabelle 60. Die Tabelle EXPLAIN_OPERATOR. PK bedeutet, dass die Spalte Teil eines Primärschlüssels ist. FK bedeutet, dass die Spalte Teil eines Fremdschlüssels ist.

Spaltenname	Datentyp	Nullwert?	Schlüssel?	Beschreibung
EXPLAIN_REQUESTER	VARCHAR(128)	Nein	FK	Berechtigungs-ID des Initiators dieser EXPLAIN-Anforderung
EXPLAIN_TIME	TIMESTAMP	Nein	FK	Initialisierungszeitpunkt für die EXPLAIN-Anforderung
SOURCE_NAME	VARCHAR(128)	Nein	FK	Name des Pakets, das ausgeführt wurde, als die dynamische Anweisung mit EXPLAIN bearbeitet wurde, oder der Name der Quellendatei, als das statische SQL mit EXPLAIN bearbeitet wurde.
SOURCE_SCHEMA	VARCHAR(128)	Nein	FK	Schema oder Qualifikationsmerkmal der Quelle der EXPLAIN-Anforderung
SOURCE_VERSION	VARCHAR(64)	Nein	FK	Version der Quelle der EXPLAIN-Anforderung
EXPLAIN_LEVEL	CHAR(1)	Nein	FK	Ebene der EXPLAIN-Informationen, für die diese Zeile relevant ist
STMTNO	INTEGER	Nein	FK	Anweisungsnummer im Paket, auf die sich diese EXPLAIN-Informationen beziehen
SECTNO	INTEGER	Nein	FK	Abschnittsnummer im Paket, auf die sich diese EXPLAIN-Informationen beziehen
OPERATOR_ID	INTEGER	Nein	Nein	Eindeutige ID für diesen Operator in dieser Abfrage
OPERATOR_TYPE	CHAR(6)	Nein	Nein	Beschreibender Kennsatz für den Operatortyp
TOTAL_COST	DOUBLE	Nein	Nein	Geschätzter kumulativer Gesamtaufwand (in Timerons) für die Ausführung des ausgewählten Zugriffsplans bis zu diesem Operator einschließlich
IO_COST	DOUBLE	Nein	Nein	Geschätzter kumulativer E/A-Aufwand (in Datenseiten-ein-/ausgaben) für die Ausführung des ausgewählten Zugriffsplans bis zu diesem Operator einschließlich
CPU_COST	DOUBLE	Nein	Nein	Geschätzter kumulativer CPU-Aufwand (in Instruktionen) für die Ausführung des ausgewählten Zugriffsplans bis zu diesem Operator einschließlich
FIRST_ROW_COST	DOUBLE	Nein	Nein	Geschätzter kumulativer Aufwand (in Timerons) für das Abrufen der ersten Zeile für den ausgewählten Zugriffsplan bis zu diesem Operator einschließlich. Dieser Wert enthält den gesamten erforderlichen Anfangsaufwand.
RE_TOTAL_COST	DOUBLE	Nein	Nein	Geschätzter kumulativer Aufwand (in Timerons) für das Abrufen der nächsten Zeile für den ausgewählten Zugriffsplan bis zu diesem Operator einschließlich
RE_IO_COST	DOUBLE	Nein	Nein	Geschätzter kumulativer E/A-Aufwand (in Datenseiten-ein-/ausgaben) für das Abrufen der nächsten Zeile für den ausgewählten Zugriffsplan bis zu diesem Operator einschließlich
RE_CPU_COST	DOUBLE	Nein	Nein	Geschätzter kumulativer CPU-Aufwand (in Instruktionen) für das Abrufen der nächsten Zeile für den ausgewählten Zugriffsplan bis zu diesem Operator einschließlich
COMM_COST	DOUBLE	Nein	Nein	Geschätzter kumulativer Übertragungsaufwand (in TCP/IP-Rahmen) für die Ausführung des ausgewählten Zugriffsplans bis zu diesem Operator einschließlich

Tabelle 60. Die Tabelle EXPLAIN_OPERATOR (Forts.). PK bedeutet, dass die Spalte Teil eines Primärschlüssels ist. FK bedeutet, dass die Spalte Teil eines Fremdschlüssels ist.

Spaltenname	Datentyp	Nullwert?	Schlüssel?	Beschreibung
FIRST_COMM_COST	DOUBLE	Nein	Nein	Geschätzter kumulativer Übertragungsaufwand (in TCP/IP-Rahmen) für das Abrufen der nächsten Zeile für den ausgewählten Zugriffsplan bis zu diesem Operator einschließlich. Dieser Wert enthält den gesamten erforderlichen Anfangsaufwand.
BUFFERS	DOUBLE	Nein	Nein	Geschätzte Puffervoraussetzungen für diesen Operator und seine Eingaben
REMOTE_TOTAL_COST	DOUBLE	Nein	Nein	Geschätzter kumulativer Gesamtaufwand (in Timerons) für die Ausführung von Operationen für die ferne(n) Datenbank(en)
REMOTE_COMM_COST	DOUBLE	Nein	Nein	Geschätzter kumulativer Übertragungsaufwand für die Ausführung des ausgewählten fernen Zugriffsplans bis zu diesem Operator einschließlich

Tabelle 61. Werte für OPERATOR_TYPE

Wert	Beschreibung
DELETE	Delete (Löschung)
FETCH	Abrufen
FILTER	Filterzeilen
GENROW	Zeile generieren
GRPBY	Gruppieren nach
HSJOIN	Hashverknüpfung
INSERT	Insert (Einfügung)
IXAND	Dynamische logische AND-Verknüpfung von Indizes über Bitzuordnungen (Index ANDing)
IXSCAN	Indexsuche
MSJOIN	Mischverknüpfung (Merge Join)
NLJOIN	Verknüpfung über Verschachtelungsschleifen (Nested Loop Join)
RETURN	Ergebnis
RIDSCN	Durchsuchen von Satz-IDs (RIDs)
SHIP	Senden der Abfrage an fernes System
SORT	Sortierung
TBSCAN	Tabellensuche
TEMP	Erstellen einer temporären Tabelle
TQ	Tabellenwarteschlange
UNION	Gesamtverknüpfung
UNIQUE	Eliminierung doppelter Werte
UPDATE	Update (Aktualisierung)

Die Tabelle EXPLAIN_PREDICATE

Die Tabelle EXPLAIN_PREDICATE gibt die Vergleichselemente an, die von einem bestimmten Operator angewandt werden.

Tabelle 62. Die Tabelle EXPLAIN_PREDICATE. PK bedeutet, dass die Spalte Teil eines Primärschlüssels ist. FK bedeutet, dass die Spalte Teil eines Fremdschlüssels ist.

Spaltenname	Datentyp	Nullwert?	Schlüssel?	Beschreibung
EXPLAIN_REQUESTER	VARCHAR(128)	Nein	FK	Berechtigungs-ID des Initiators dieser EXPLAIN-Anforderung
EXPLAIN_TIME	TIMESTAMP	Nein	FK	Initialisierungszeitpunkt für die EXPLAIN-Anforderung
SOURCE_NAME	VARCHAR(128)	Nein	FK	Name des Pakets, das ausgeführt wurde, als die dynamische Anweisung mit EXPLAIN bearbeitet wurde, oder der Name der Quellendatei, als das statische SQL mit EXPLAIN bearbeitet wurde.
SOURCE_SCHEMA	VARCHAR(128)	Nein	FK	Schema oder Qualifikationsmerkmal der Quelle der EXPLAIN-Anforderung
SOURCE_VERSION	VARCHAR(64)	Nein	FK	Version der Quelle der EXPLAIN-Anforderung
EXPLAIN_LEVEL	CHAR(1)	Nein	FK	Ebene der EXPLAIN-Informationen, für die diese Zeile relevant ist
STMTNO	INTEGER	Nein	FK	Anweisungsnummer im Paket, auf die sich diese EXPLAIN-Informationen beziehen
SECTNO	INTEGER	Nein	FK	Abschnittsnummer im Paket, auf die sich diese EXPLAIN-Informationen beziehen
OPERATOR_ID	INTEGER	Nein	Nein	Eindeutige ID für diesen Operator in dieser Abfrage
PREDICATE_ID	INTEGER	Nein	Nein	Eindeutige ID für dieses Vergleichselement für den angegebenen Operator
HOW_APPLIED	CHAR(5)	Nein	Nein	Verwendungsart des angegebenen Operators für dieses Vergleichselement
WHEN_EVALUATED	CHAR(3)	Nein	Nein	Kennzeichnet, wann die in diesem Vergleichselement verwendete Unterabfrage ausgewertet wird. Die folgenden Werte sind möglich: **Leer** Dieses Vergleichselement enthält keine Unterabfrage. **EAA** Die in diesem Vergleichselement verwendete Unterabfrage wird bei der Anwendung des Vergleichselement (EAA) ausgewertet. D. h., sie wird bei der Anwendung des Vergleichselements für jede vom angegebenen Operator verarbeiteten Zeile erneut ausgewertet. **EAO** Die in diesem Vergleichselement verwendete Unterabfrage wird beim Öffnen des Vergleichselements (EAO) ausgewertet. D. h., sie wird für den angegebenen Operator nur einmal erneut ausgewertet, und ihre Ergebnisse werden bei der Anwendung des Vergleichselements für jede Zeile erneut verwendet. **MUL** Es gibt mehrere Arten von Unterabfragen in diesem Vergleichselement.
RELOP_TYPE	CHAR(2)	Nein	Nein	Der Vergleichsoperatortyp, der in diesem Vergleichselement verwendet wird

Tabelle 62. Die Tabelle EXPLAIN_PREDICATE (Forts.). PK bedeutet, dass die Spalte Teil eines Primärschlüssels ist. FK bedeutet, dass die Spalte Teil eines Fremdschlüssels ist.

Spaltenname	Datentyp	Nullwert?	Schlüssel?	Beschreibung
SUBQUERY	CHAR(1)	Nein	Nein	Gibt an, ob ein Datenstrom aus einer Unterabfrage für dieses Vergleichselement erforderlich ist. Es können mehrere Datenströme aus Unterabfragen erforderlich sein. Die folgenden Werte sind möglich: **N** Es ist kein Datenstrom aus einer Unterabfrage erforderlich. **Y** Ein oder mehrere Datenströme sind aus einer Unterabfrage erforderlich.
FILTER_FACTOR	DOUBLE	Nein	Nein	Der geschätzte Anteil der Zeilen, die durch dieses Vergleichselement qualifiziert wird
PREDICATE_TEXT	CLOB(2M)	Ja	Nein	Der Text des Vergleichselements wie aus der internen Darstellung der SQL-Anweisung erneut erstellt. Diese Spalte zeichnet auch den Wert der Hostvariablen, des Sonderregisters oder der Parametermarke auf, falls ein solches Element bei der Kompilierung der Anweisung verwendet wird. Der Wert wird nur in die Tabelle EXPLAIN_PREDICATE ausgelesen, wenn die Anweisung von einem Benutzer ausgeführt wird, der die Berechtigung DBADM besitzt. Beachten Sie, dass andere Benutzer ohne DBADM-Berechtigung diesen Wert zwar nicht in diese Tabelle auslesen können, aber trotzdem die Möglichkeit haben können, ihn anzuzeigen, wenn er sich bereits in der Tabelle befindet. Null, wenn nicht vorhanden.

Tabelle 63. Mögliche Werte für HOW_APPLIED

Wert	Beschreibung
BSARG	Wird für jeden Block als ein als Suchargument verwendbares Vergleichselement ausgewertet.
JOIN	Zum Verknüpfen von Tabellen verwendet
RESID	Als Restvergleichselement ausgewertet
SARG	Wird ausgewertet als ein als Suchargument verwendbares Vergleichselement für Index oder Datenseite.
START	Als Startbedingung verwendet
STOP	Als Stoppbedingung verwendet

Tabelle 64. Gültige Werte für RELOP_TYPE

Wert	Beschreibung
Leerzeichen	Nicht zutreffend
EQ	Gleich
GE	Größer-gleich
GT	Größer als
IN	IN-Liste
LE	Kleiner-gleich
LK	Ähnlich
LT	Kleiner als
NE	Ungleich
NL	Gleich null
NN	Ungleich null

Die Tabelle EXPLAIN_STATEMENT

Die Tabelle EXPLAIN_STATEMENT enthält den Text der SQL-Anweisung, wie sie für die verschiedenen Stufen der EXPLAIN-Informationen existiert. Die ursprüngliche SQL-Anweisung, wie sie vom Benutzer eingegeben wurde, wird in dieser Tabelle zusammen mit der (vom Optimierungsprogramm) zum Auswählen eines Zugriffsplans für die Ausführung der SQL-Anweisung verwendeten Version gespeichert. Die letztere Version hat oft sehr wenig Ähnlichkeit mit dem Original, da sie möglicherweise vom SQL-Compiler umgeschrieben und/oder mit zusätzlichen Vergleichselementen versehen wurde.

Tabelle 65. Tabelle EXPLAIN_STATEMENT. PK bedeutet, dass die Spalte Teil eines Primärschlüssels ist. FK bedeutet, dass die Spalte Teil eines Fremdschlüssels ist.

Spaltenname	Datentyp	Nullwert?	Schlüssel?	Beschreibung
EXPLAIN_REQUESTER	VARCHAR(128)	Nein	PK, FK	Berechtigungs-ID des Initiators dieser EXPLAIN-Anforderung
EXPLAIN_TIME	TIMESTAMP	Nein	PK, FK	Initialisierungszeitpunkt für die EXPLAIN-Anforderung
SOURCE_NAME	VARCHAR(128)	Nein	PK, FK	Name des Pakets, das ausgeführt wurde, als die dynamische Anweisung mit EXPLAIN bearbeitet wurde, oder der Name der Quellendatei, als das statische SQL mit EXPLAIN bearbeitet wurde.
SOURCE_SCHEMA	VARCHAR(128)	Nein	PK, FK	Schema oder Qualifikationsmerkmal der Quelle der EXPLAIN-Anforderung
SOURCE_VERSION	VARCHAR(64)	Nein	FK	Version der Quelle der EXPLAIN-Anforderung
EXPLAIN_LEVEL	CHAR(1)	Nein	PK	Ebene der EXPLAIN-Informationen, für die diese Zeile relevant ist Gültige Werte: **O** Originaltext (wie vom Benutzer eingegeben) **P** PLANAUSWAHL
STMTNO	INTEGER	Nein	PK	Anweisungsnummer im Paket, auf die sich diese EXPLAIN-Informationen beziehen. Bei dynamischen EXPLAIN-SQL-Anweisungen wird dieser Wert auf 1 gesetzt. Bei statischen SQL-Anweisungen gleicht dieser Wert dem für die Katalogsicht SYSCAT.STATEMENTS verwendeten Wert.
SECTNO	INTEGER	Nein	PK	Abschnittsnummer in dem Paket, die diese SQL-Anweisung enthält. Bei dynamischen SQL-Anweisungen EXPLAIN ist dies die Abschnittsnummer, die zum Speichern des Abschnitts für diese Anweisung zur Laufzeit verwendet wird. Bei statischen SQL-Anweisungen gleicht dieser Wert dem für die Katalogsicht SYSCAT.STATEMENTS verwendeten Wert.
QUERYNO	INTEGER	Nein	Nein	Numerische Kennung für eine mit EXPLAIN bearbeitete SQL-Anweisung. Für dynamische SQL-Anweisungen (außer der SQL-Anweisung EXPLAIN), die mit Hilfe von CLP oder CLI angegeben wurden, ist der Standardwert ein sequenziell inkrementierter Wert. Andernfalls ist der Standardwert für statische SQL-Anweisungen der Wert STMTNO und für dynamische SQL-Anweisungen der Wert 1.
QUERYTAG	CHAR(20)	Nein	Nein	Identifizierendes Kennzeichen für jede mit EXPLAIN bearbeitete SQL-Anweisung. Für dynamische SQL-Anweisungen, die mit CLP angegeben wurden (außer der SQL-Anweisung EXPLAIN), ist der Standardwert 'CLP'. Für dynamische SQL-Anweisungen, die mit CLI angegeben wurden (außer der SQL-Anweisung EXPLAIN), ist der Standardwert 'CLI'. Andernfalls werden Leerzeichen als Standardwert verwendet.

Tabelle 65. Tabelle EXPLAIN_STATEMENT (Forts.). PK bedeutet, dass die Spalte Teil eines Primärschlüssels ist. FK bedeutet, dass die Spalte Teil eines Fremdschlüssels ist.

Spaltenname	Datentyp	Nullwert?	Schlüssel?	Beschreibung
STATEMENT_TYPE	CHAR(2)	Nein	Nein	Beschreibender Kennsatz für den mit EXPLAIN bearbeiteten Abfragetyp. Die folgenden Werte sind möglich: **S** Auswählen **D** Löschen **DC** Löschen bei aktueller Cursorposition **I** Einfügen **U** Aktualisieren **UC** Aktualisieren bei aktueller Cursorposition
UPDATABLE	CHAR(1)	Nein	Nein	Zeigt an, ob diese Anweisung als aktualisierbar angesehen wird. Dies ist besonders wichtig für Anweisungen SELECT, die als potenziell aktualisierbar bestimmt werden können. Die folgenden Werte sind möglich: **' '** Nicht zutreffend (leer) **N** Nein **Y** Ja
DELETABLE	CHAR(1)	Nein	Nein	Zeigt an, ob diese Anweisung als löschbar angesehen wird. Dies ist besonders wichtig für Anweisungen SELECT, die als potenziell löschbar bestimmt werden können. Die folgenden Werte sind möglich: **' '** Nicht zutreffend (leer) **N** Nein **Y** Ja
TOTAL_COST	DOUBLE	Nein	Nein	Der geschätzte Gesamtaufwand (in Timerons) der Ausführung des ausgewählten Zugriffsplans für diese Anweisung; der Wert wird auf 0 (null) gesetzt, wenn EXPLAIN_LEVEL *O* ist (ursprünglicher Text), da zu diesem Zeitpunkt kein Zugriffsplan ausgewählt ist.
STATEMENT_TEXT	CLOB(2M)	Nein	Nein	Text oder Teil des Textes der mit EXPLAIN bearbeiteten SQL-Anweisung. Der Text, der für die Planauswahlstufe von EXPLAIN gezeigt wird, wurde aus der internen Darstellung rekonstruiert und ist SQL-ähnlich; das bedeutet, dass nicht garantiert wird, dass die rekonstruierte Anweisung der gültigen SQL-Syntax entspricht.
SNAPSHOT	BLOB(10M)	Ja	Nein	Momentaufnahme der internen Darstellung für diese SQL-Anweisung beim angezeigten EXPLAIN_LEVEL. Diese Spalte ist für die Verwendung mit DB2 Visual Explain konzipiert. Die Spalte wird auf null gesetzt, wenn der Wert für EXPLAIN_LEVEL gleich *0* ist (ursprüngliche Anweisung), da zu dem Zeitpunkt, zu dem diese spezifische Version der Anweisung erfasst wird, kein Zugriffsplan ausgewählt ist.
QUERY_DEGREE	INTEGER	Nein	Nein	Gibt den Grad der partitionsinternen Parallelität zum Zeitpunkt des Aufrufs von EXPLAIN an. Für die Originalanweisung enthält diese Angabe den gesteuerten Grad der partitionsinternem Parallelität. Bei PLANAUSWAHL enthält sie den Grad der partitionsinternen Parallelität, der für den zu verwendenden Zugriffsplan generiert wurde.

Tabelle 65. Tabelle EXPLAIN_STATEMENT (Forts.). PK bedeutet, dass die Spalte Teil eines Primärschlüssels ist. FK bedeutet, dass die Spalte Teil eines Fremdschlüssels ist.

Spaltenname	Datentyp	Nullwert?	Schlüssel?	Beschreibung
REOPT	CHAR(1)	Nein	Nein	Gibt an, ob die SQL-Anweisung mit realen Werten für die Hostvariablen, Parametermarken oder Sonderregister reoptimiert wird oder nicht. Mögliche Werte: **N** Erstellen eines Zugriffspfads mit Standardschätzwerten für die Variablen **Y** Reoptimieren des Zugriffspfads mit verfügbaren Werten für die Hostvariablen, Parametermarken oder Sonderregister

Die Tabelle EXPLAIN_STREAM

Die Tabelle EXPLAIN_STREAM zeigt den Eingabe- und Ausgabedatenstrom zwischen einzelnen Operatoren und Datenobjekten. Die Datenobjekte selbst werden in der Tabelle EXPLAIN_OBJECT beschrieben. Die an einem Datenstrom beteiligten Operatoren befinden sich in der Tabelle EXPLAIN_OPERATOR.

Tabelle 66. Die Tabelle EXPLAIN_STREAM. PK bedeutet, dass die Spalte Teil eines Primärschlüssels ist. FK bedeutet, dass die Spalte Teil eines Fremdschlüssels ist.

Spaltenname	Datentyp	Nullwert?	Schlüssel?	Beschreibung
EXPLAIN_REQUESTER	VARCHAR(128)	Nein	FK	Berechtigungs-ID des Initiators dieser EXPLAIN-Anforderung
EXPLAIN_TIME	TIMESTAMP	Nein	FK	Initialisierungszeitpunkt für die EXPLAIN-Anforderung
SOURCE_NAME	VARCHAR(128)	Nein	FK	Name des Pakets, das ausgeführt wurde, als die dynamische Anweisung mit EXPLAIN bearbeitet wurde, oder der Name der Quellendatei, als das statische SQL mit EXPLAIN bearbeitet wurde.
SOURCE_SCHEMA	VARCHAR(128)	Nein	FK	Schema oder Qualifikationsmerkmal der Quelle der EXPLAIN-Anforderung
SOURCE_VERSION	VARCHAR(64)	Nein	FK	Version der Quelle der EXPLAIN-Anforderung
EXPLAIN_LEVEL	CHAR(1)	Nein	FK	Ebene der EXPLAIN-Informationen, für die diese Zeile relevant ist
STMTNO	INTEGER	Nein	FK	Anweisungsnummer im Paket, auf die sich diese EXPLAIN-Informationen beziehen
SECTNO	INTEGER	Nein	FK	Abschnittsnummer im Paket, auf die sich diese EXPLAIN-Informationen beziehen
STREAM_ID	INTEGER	Nein	Nein	Eindeutige ID für diesen Datenstrom im angegebenen Operator
SOURCE_TYPE	CHAR(1)	Nein	Nein	Kennzeichnet die Quelle dieses Datenstroms: **O** Operator **D** Datenobjekt
SOURCE_ID	SMALLINT	Nein	Nein	Eindeutige ID für den Operator in dieser Abfrage, die die Quelle dieses Datenstroms ist. Wird auf -1 gesetzt, wenn SOURCE_TYPE 'D' ist.
TARGET_TYPE	CHAR(1)	Nein	Nein	Kennzeichnet das Ziel dieses Datenstroms: **O** Operator **D** Datenobjekt
TARGET_ID	SMALLINT	Nein	Nein	Eindeutige ID für den Operator in dieser Abfrage, die das Ziel dieses Datenstroms ist. Wird auf -1 gesetzt, wenn TARGET_TYPE 'D' ist.
OBJECT_SCHEMA	VARCHAR(128)	Ja	Nein	Schema, zu dem das betreffende Datenobjekt gehört. Wird auf null gesetzt, wenn SOURCE_TYPE und TARGET_TYPE gleich 'O' sind.
OBJECT_NAME	VARCHAR(128)	Ja	Nein	Name des Objekts, das das Subjekt des Datenstroms ist. Wird auf null gesetzt, wenn SOURCE_TYPE und TARGET_TYPE gleich 'O' sind.
STREAM_COUNT	DOUBLE	Nein	Nein	Geschätzte Kardinalität des Datenstroms
COLUMN_COUNT	SMALLINT	Nein	Nein	Anzahl der Spalten im Datenstrom
PREDICATE_ID	INTEGER	Nein	Nein	Wenn dieser Datenstrom Teil einer Unterabfrage für ein Vergleichselement ist, wird hier die Vergleichselement-ID wiedergegeben; andernfalls wird die Spalte auf -1 gesetzt.

Tabelle EXPLAIN_STREAM

Tabelle 66. Die Tabelle EXPLAIN_STREAM (Forts.). PK bedeutet, dass die Spalte Teil eines Primärschlüssels ist. FK bedeutet, dass die Spalte Teil eines Fremdschlüssels ist.

Spaltenname	Datentyp	Nullwert?	Schlüssel?	Beschreibung
COLUMN_NAMES	CLOB(2M)	Ja	Nein	Diese Spalte enthält die Namen und die Informationen zur Reihenfolge der von diesem Datenstrom betroffenen Spalten. Diese Namen haben folgendes Format: `NAME1(A)+NAME2(D)+NAME3+NAME4` Dabei gibt *(A)* eine aufsteigend sortierte Spalte an, *(D)* gibt eine absteigend sortierte Spalte an, und fehlende Sortierangaben bedeuten, dass die Spalte entweder nicht sortiert oder die Sortierfolge nicht relevant ist.
PMID	SMALLINT	Nein	Nein	ID der Partitionierungszuordnung
SINGLE_NODE	CHAR(5)	Ja	Nein	Gibt an, ob dieser Datenstrom sich auf einer einzigen oder auf mehreren Partitionen befindet: **MULT** Auf mehreren Partitionen **COOR** Auf Koordinatorknoten **HASH** Weiterleitung durch Hashverfahren **RID** Weiterleitung durch Zeilen-ID **FUNC** Weiterleitung durch Funktion (HASHEDVALUE() oder DBPARTITIONNUM()) **CORR** Weiterleitung durch Korrelationswert **Numerisch** Auf vorbestimmten einzigen Knoten gerichtet
PARTITION_COLUMNS	CLOB(2M)	Ja	Nein	Liste der Spalten, über die dieser Datenstrom partitioniert ist

Die Tabelle ADVISE_INDEX

Die Tabelle ADVISE_INDEX zeigt die empfohlenen Indizes.

Tabelle 67. Die Tabelle ADVISE_INDEX. PK bedeutet, dass die Spalte Teil eines Primärschlüssels ist. FK bedeutet, dass die Spalte Teil eines Fremdschlüssels ist.

Spaltenname	Datentyp	Nullwert?	Schlüssel?	Beschreibung
EXPLAIN_REQUESTER	VARCHAR(128)	Nein	Nein	Berechtigungs-ID des Initiators dieser EXPLAIN-Anforderung
EXPLAIN_TIME	TIMESTAMP	Nein	Nein	Initialisierungszeitpunkt für die EXPLAIN-Anforderung
SOURCE_NAME	VARCHAR(128)	Nein	Nein	Name des Pakets, das ausgeführt wurde, als die dynamische Anweisung mit EXPLAIN bearbeitet wurde, oder der Name der Quellendatei, als das statische SQL mit EXPLAIN bearbeitet wurde.
SOURCE_SCHEMA	VARCHAR(128)	Nein	Nein	Schema oder Qualifikationsmerkmal der Quelle der EXPLAIN-Anforderung
SOURCE_VERSION	VARCHAR(64)	Nein	Nein	Version der Quelle der EXPLAIN-Anforderung
EXPLAIN_LEVEL	CHAR(1)	Nein	Nein	Ebene der EXPLAIN-Informationen, für die diese Zeile relevant ist
STMTNO	INTEGER	Nein	Nein	Anweisungsnummer im Paket, auf die sich diese EXPLAIN-Informationen beziehen
SECTNO	INTEGER	Nein	Nein	Abschnittsnummer im Paket, auf die sich diese EXPLAIN-Informationen beziehen
QUERYNO	INTEGER	Nein	Nein	Numerische Kennung für eine mit EXPLAIN bearbeitete SQL-Anweisung. Für dynamische SQL-Anweisungen (außer der SQL-Anweisung EXPLAIN), die mit Hilfe von CLP oder CLI angegeben wurden, ist der Standardwert ein sequenziell inkrementierter Wert. Andernfalls ist der Standardwert für statische SQL-Anweisungen der Wert STMTNO und für dynamische SQL-Anweisungen der Wert 1.
QUERYTAG	CHAR(20)	Nein	Nein	Identifizierendes Kennzeichen für jede mit EXPLAIN bearbeitet SQL-Anweisung. Für dynamische SQL-Anweisungen, die mit CLP angegeben wurden (außer der EXPLAIN-SQL-Anweisung), ist der Standardwert 'CLP'. Für dynamische SQL-Anweisungen, die mit CLI angegeben wurden (außer der EXPLAIN-SQL-Anweisung), ist der Standardwert 'CLI'. Andernfalls werden Leerzeichen als Standardwert verwendet.
NAME	VARCHAR(128)	Nein	Nein	Name des Index.
CREATOR	VARCHAR(128)	Nein	Nein	Qualifikationsmerkmal des Indexnamens.
TBNAME	VARCHAR(128)	Nein	Nein	Name oder Kurzname der Tabelle, für die der Index definiert ist
TBCREATOR	VARCHAR(128)	Nein	Nein	Qualifikationsmerkmal des Tabellennamens
COLNAMES	CLOB(2M)	Nein	Nein	Liste mit Spaltennamen
UNIQUERULE	CHAR(1)	Nein	Nein	Regel zur Eindeutigkeit: D=Gleiche Werte zulässig P=Primärindex U=Nur eindeutige Einträge zulässig
COLCOUNT	SMALLINT	Nein	Nein	Anzahl der Spalten im Schlüssel plus Anzahl der INCLUDE-Spalten, wenn vorhanden
IID	SMALLINT	Nein	Nein	Interne Index-ID
NLEAF	INTEGER	Nein	Nein	Anzahl der Blattseiten (Leaf pages); –1, wenn keine statistischen Daten erfasst wurden.

Tabelle 67. Die Tabelle ADVISE_INDEX (Forts.). PK bedeutet, dass die Spalte Teil eines Primärschlüssels ist. FK bedeutet, dass die Spalte Teil eines Fremdschlüssels ist.

Spaltenname	Datentyp	Nullwert?	Schlüssel?	Beschreibung
NLEVELS	SMALLINT	Nein	Nein	Anzahl der Indexstufen; –1, wenn keine statistischen Daten erfasst wurden.
FIRSTKEYCARD	BIGINT	Nein	Nein	Anzahl der eindeutigen ersten Schlüsselwerte; –1, wenn keine statistischen Daten erfasst wurden.
FULLKEYCARD	BIGINT	Nein	Nein	Anzahl der eindeutigen vollständigen Schlüsselwerte; –1, wenn keine statistischen Daten erfasst wurden.
CLUSTERRATIO	SMALLINT	Nein	Nein	Grad der Datenclusterbildung bezüglich des Index; –1, wenn keine statistischen Daten erfasst wurden oder wenn detaillierte statistische Daten zum Index erfasst wurden (in diesem Fall wird CLUSTERFACTOR verwendet).
CLUSTERFACTOR	DOUBLE	Nein	Nein	Feinere Erfassung des Grades der Clusterbildung oder –1, wenn keine detaillierte Indexstatistik gesammelt wurde oder wenn der Index für einen Kurznamen definiert ist.
USERDEFINED	SMALLINT	Nein	Nein	Vom Benutzer definiert.
SYSTEM_REQUIRED	SMALLINT	Nein	Nein	1, wenn die eine oder die andere der folgenden Bedingungen gilt: – Dieser Index ist für eine Integritätsbedingung durch einen Primärschlüssel oder eindeutigen Schlüssel erforderlich oder dieser Index ist ein Dimensionsblockindex bzw. zusammengesetzter Blockindex für eine MDC-Tabelle (Multi-Dimensional Clustering). – Dies ist ein Index für die Spalte (OID) einer typisierten Tabelle. 2, falls beide der folgenden Bedingungen gelten: – Dieser Index ist für eine Integritätsbedingung durch einen Primärschlüssel oder eindeutigen Schlüssel erforderlich oder dieser Index ist ein Dimensionsblockindex bzw. zusammengesetzter Blockindex für eine MDC-Tabelle. – Dies ist ein Index für die Spalte (OID) einer typisierten Tabelle. Ansonsten 0.
CREATE_TIME	TIMESTAMP	Nein	Nein	Zeitmarke für den Zeitpunkt der Indexerstellung
STATS_TIME	TIMESTAMP	Ja	Nein	Zeitmarke für den Zeitpunkt der letzten Änderung an den für diesen Index aufgezeichneten Statistikdaten. Null, wenn keine Statistikdaten verfügbar sind.
PAGE_FETCH_PAIRS	VARCHAR(254)	Nein	Nein	Eine Liste von Paaren ganzer Zahlen, die in Zeichenform dargestellt sind. Jedes Paar zeigt die Anzahl der Seiten in einem angenommenen Puffer, und die Anzahl der Abrufoperationen für Seiten (Page fetches), die erforderlich wäre, um die Tabelle mit diesem Index unter Verwendung des angenommenen Puffers zu durchsuchen. (Eine Zeichenfolge der Länge 0, wenn keine Daten verfügbar sind.)
REMARKS	VARCHAR(254)	Ja	Nein	Ein Kommentar des Benutzers oder null
DEFINER	VARCHAR(128)	Nein	Nein	Der Benutzer, der den Index erstellt hat
CONVERTED	CHAR(1)	Nein	Nein	Zur zukünftigen Verwendung reserviert.
SEQUENTIAL_PAGES	INTEGER	Nein	Nein	Anzahl der Blattseiten (Leaf pages), die sich mit wenigen oder keinen größeren Abständen nach Indexschlüsseln sortiert auf der Platte befinden. (–1, wenn keine statistischen Daten verfügbar sind.)

Tabelle 67. Die Tabelle ADVISE_INDEX (Forts.). PK bedeutet, dass die Spalte Teil eines Primärschlüssels ist. FK bedeutet, dass die Spalte Teil eines Fremdschlüssels ist.

Spaltenname	Datentyp	Nullwert?	Schlüssel?	Beschreibung
DENSITY	INTEGER	Nein	Nein	Verhältnis der SEQUENTIAL_PAGES zur Anzahl der Indexseiten ausgedrückt als Prozentwert (ganze Zahl zwischen 0 und 100, –1, wenn keine statistischen Daten verfügbar sind).
FIRST2KEYCARD	BIGINT	Nein	Nein	Anzahl der eindeutigen Schlüssel in den ersten beiden Spalten des Index (–1, wenn keine statistischen Daten gesammelt werden oder wenn dies nicht zutrifft)
FIRST3KEYCARD	BIGINT	Nein	Nein	Anzahl der eindeutigen Schlüssel in den ersten drei Spalten des Index (–1, wenn keine statistischen Daten gesammelt werden oder wenn dies nicht zutrifft)
FIRST4KEYCARD	BIGINT	Nein	Nein	Anzahl der eindeutigen Schlüssel in den ersten vier Spalten des Index (–1, wenn keine statistischen Daten gesammelt werden oder wenn dies nicht zutrifft)
PCTFREE	SMALLINT	Nein	Nein	Prozentsatz jeder äußeren Seite (Blattseite) des Index, der beim ersten Erstellen des Index reserviert wird. Dieser Speicherbereich ist für zukünftige Einfügungen nach der Erstellung des Index verfügbar.
UNIQUE_COLCOUNT	SMALLINT	Nein	Nein	Anzahl der für einen eindeutigen Schlüssel erforderlichen Spalten. Immer <=COLCOUNT. < COLCOUNT nur, wenn INCLUDE-Spalten vorhanden sind. –1, wenn der Index keinen eindeutigen Schlüssel hat (gleiche Werte möglich).
MINPCTUSED	SMALLINT	Nein	Nein	Wenn nicht Null, ist die Onlinedefragmentierung des Index aktiviert, und der Wert ist der Schwellenwert des minimal belegten Speicherbereichs vor dem Zusammenfügen der Seiten.
REVERSE_SCANS	CHAR(1)	Nein	Nein	Y=Index unterstützt umgekehrtes Durchsuchen N=Index unterstützt nicht umgekehrtes Durchsuchen
USE_INDEX	CHAR(1)	Ja	Nein	Y=Index empfohlen oder ausgewertet N=Index wird nicht empfohlen
CREATION_TEXT	CLOB(2M)	Nein	Nein	Die zum Erstellen des Index verwendete SQL-Anweisung.
PACKED_DESC	BLOB(1M)	Ja	Nein	Interne Beschreibung der Tabelle

Die Tabelle ADVISE_INSTANCE

Die Tabelle ADVISE_INSTANCE enthält Informationen zur Ausführung von **db2advis**. Diese Informationen beinhalten auch die Startzeit. Für jede Ausführung von **db2advis** wird eine Zeile eingefügt. Andere ADVISE-Tabellen besitzen einen Fremdschlüssel (RUN_ID), der mit der Spalte START_TIME der Tabelle ADVISE_INSTANCE verbunden ist, und enthalten Zeilen, die während der gleichen Ausführung des Designadvisors erstellt wurden.

Tabelle 68. Tabelle ADVISE_INSTANCE. PK bedeutet, dass die Spalte Teil eines Primärschlüssels ist. FK bedeutet, dass die Spalte Teil eines Fremdschlüssels ist.

Spaltenname	Datentyp	Nullwert?	Schlüssel?	Beschreibung
START_TIME	TIMESTAMP	Nein	PK	Der Zeitpunkt, zu dem die Ausführung von **db2advis** beginnt.
END_TIME	TIMESTAMP	Nein	Nein	Der Zeitpunkt, zu dem die Ausführung von **db2advis** endet.
MODE	VARCHAR(4)	Nein	Nein	Der Wert, der mit der Option -m für den Designadvisor angegeben wurde, zum Beispiel 'MC' zur Angabe von MQT und MDC.
WKLD_COMPRESSION	CHAR(4)	Nein	Nein	Die Auslastungskomprimierung, unter der der Designadvisor ausgeführt wurde.
STATUS	CHAR(9)	Nein	Nein	Der Status einer Ausführung des Designadvisors. Der Status kann die Werte 'STARTED' oder 'COMPLETED' (wenn erfolgreich) annehmen bzw. eine Fehlernummer mit dem Präfix 'EI' für internen Fehler oder 'EX' für externen Fehler sein, wobei die Fehlernummer den SQLCODE darstellt.

Die Tabelle ADVISE_MQT

Die Tabelle ADVISE_MQT enthält Informationen zu gespeicherten Abfragetabellen (MQTs), die vom Designadvisor empfohlen werden.

Tabelle 69. Tabelle ADVISE_MQT. PK bedeutet, dass die Spalte Teil eines Primärschlüssels ist. FK bedeutet, dass die Spalte Teil eines Fremdschlüssels ist.

Spaltenname	Datentyp	Nullwert?	Schlüssel?	Beschreibung
EXPLAIN_REQUESTER	VARCHAR(128)	Nein	Nein	Berechtigungs-ID des Initiators dieser EXPLAIN-Anforderung
EXPLAIN_TIME	TIMESTAMP	Nein	Nein	Initialisierungszeitpunkt für die EXPLAIN-Anforderung
SOURCE_NAME	VARCHAR(128)	Nein	Nein	Name des Pakets, das ausgeführt wurde, als die dynamische Anweisung mit EXPLAIN bearbeitet wurde, oder der Name der Quellendatei, als das statische SQL mit EXPLAIN bearbeitet wurde.
SOURCE_SCHEMA	VARCHAR(128)	Nein	Nein	Schema oder Qualifikationsmerkmal der Quelle der EXPLAIN-Anforderung
SOURCE_VERSION	VARCHAR(64)	Nein	Nein	Version der Quelle der EXPLAIN-Anforderung
EXPLAIN_LEVEL	CHAR(1)	Nein	Nein	Ebene der EXPLAIN-Informationen, für die diese Zeile relevant ist
STMTNO	INTEGER	Nein	Nein	Anweisungsnummer im Paket, auf die sich diese EXPLAIN-Informationen beziehen
SECTNO	INTEGER	Nein	Nein	Anweisungsnummer im Paket, auf die sich diese EXPLAIN-Informationen beziehen
NAME	VARCHAR(128)	Nein	Nein	MQT-Name
CREATOR	VARCHAR(128)	Nein	Nein	Name des Erstellers der MQT
IID	SMALLINT	Nein	Nein	Interne Kennung
CREATE_TIME	TIMESTAMP	Nein	Nein	Zeitmarke für den Zeitpunkt der Erstellung der MQT
STATS_TIME	TIMESTAMP	Ja	Nein	Zeitmarke für den Zeitpunkt der Statistikerfassung
NUMROWS	DOUBLE	Nein	Nein	Die Anzahl der geschätzten Zeilen in der MQT
NUMCOLS	SMALLINT	Nein	Nein	Anzahl der in der MQT definierten Spalten
ROWSIZE	DOUBLE	Nein	Nein	Durchschnittliche Länge (in Byte) einer Zeile in der MQT
BENEFIT	FLOAT	Nein	Nein	Zur zukünftigen Verwendung reserviert.
USE_MQT	CHAR(1)	Ja	Nein	Auf 'Y' gesetzt, wenn die MQT empfohlen wird.
MQT_SOURCE	CHAR(1)	Ja	Nein	Gibt an, wo der MQT-Kandidat generiert wurde. Auf 'I' gesetzt, wenn der MQT-Kandidat eine MQT mit REFRESH IMMEDIATE ist, oder auf 'D' gesetzt, wenn die MQT nur mit REFRESH DEFERRED erstellt werden kann.
QUERY_TEXT	CLOB(2M)	Nein	Nein	Enthält die Abfrage, die die MQT definiert.
CREATION_TEXT	CLOB(2M)	Nein	Nein	Enthält die DDL-Anweisungen CREATE TABLE für die MQT.
SAMPLE_TEXT	CLOB(2M)	Nein	Nein	Enthält die Stichprobenabfrage, die zum Abrufen detaillierter Statistiken für die MQT verwendet wird. Wird nur verwendet, wenn für den Designadvisor detaillierte Statistiken erforderlich sind. Die resultierenden Stichprobenstatistiken werden in dieser Tabelle gezeigt. Wenn null, wurde keine Stichprobenabfrage für diese MQT erstellt.
COLSTATS	CLOB(2M)	Nein	Nein	Enthält die Spaltenstatistiken für die MQT (falls nicht null). Diese Statistiken liegen im XML-Format vor und enthalten den Spaltennamen, die Spaltenkardinalität und optional die Werte für HIGH2KEY und LOW2KEY.

Tabelle ADVISE_MQT

Tabelle 69. Tabelle ADVISE_MQT (Forts.). PK bedeutet, dass die Spalte Teil eines Primärschlüssels ist. FK bedeutet, dass die Spalte Teil eines Fremdschlüssels ist.

Spaltenname	Datentyp	Nullwert?	Schlüssel?	Beschreibung
EXTRA_INFO	BLOB(2M)	Nein	Nein	Reserviert für verschiedene Ausgaben.
TBSPACE	VARCHAR(128)	Nein	Nein	Der für die MQT empfohlene Tabellenbereich
RUN_ID	TIMESTAMP	Ja	FK	Ein Wert, der dem Wert für START_TIME einer Zeile in der Tabelle ADVISE_INSTANCE entspricht, so dass diese Zeile der gleichen Ausführung des Designadvisors zugeordnet wird.
REFRESH_TYPE	CHAR(1)	Nein	Nein	Entweder 'I' für IMMEDIATE oder 'D' für DEFERRED.
EXISTS	CHAR(1)	Nein	Nein	Auf 'Y' gesetzt, wenn die MQT im Datenbankkatalog vorhanden ist.

Die Tabelle ADVISE_PARTITION

Die Tabelle ADVISE_PARTITION enthält Informationen zu Datenbankpartitionen, die vom Designadvisor empfohlen werden. Sie kann nur in einer partitionierten Datenbankumgebung mit Daten gefüllt werden.

Tabelle 70. Tabelle ADVISE_PARTITION. PK bedeutet, dass die Spalte Teil eines Primärschlüssels ist. FK bedeutet, dass die Spalte Teil eines Fremdschlüssels ist.

Spaltenname	Datentyp	Nullwert?	Schlüssel?	Beschreibung
EXPLAIN_REQUESTER	VARCHAR(128)	Nein	Nein	Berechtigungs-ID des Initiators dieser EXPLAIN-Anforderung
EXPLAIN_TIME	TIMESTAMP	Nein	Nein	Initialisierungszeitpunkt für die EXPLAIN-Anforderung
SOURCE_NAME	VARCHAR(128)	Nein	Nein	Name des Pakets, das ausgeführt wurde, als die dynamische Anweisung mit EXPLAIN bearbeitet wurde, oder der Name der Quellendatei, als das statische SQL mit EXPLAIN bearbeitet wurde.
SOURCE_SCHEMA	VARCHAR(128)	Nein	Nein	Schema oder Qualifikationsmerkmal der Quelle der EXPLAIN-Anforderung
SOURCE_VERSION	VARCHAR(64)	Nein	Nein	Version der Quelle der EXPLAIN-Anforderung
EXPLAIN_LEVEL	CHAR(1)	Nein	Nein	Ebene der EXPLAIN-Informationen, für die diese Zeile relevant ist
STMTNO	INTEGER	Nein	Nein	Anweisungsnummer im Paket, auf die sich diese EXPLAIN-Informationen beziehen
SECTNO	INTEGER	Nein	Nein	Anweisungsnummer im Paket, auf die sich diese EXPLAIN-Informationen beziehen
QUERYNO	INTEGER	Nein	Nein	Numerische Kennung für eine mit EXPLAIN bearbeitete SQL-Anweisung. Für dynamische SQL-Anweisungen (außer der SQL-Anweisung EXPLAIN), die mit Hilfe von CLP oder CLI angegeben wurden, ist der Standardwert ein sequenziell inkrementierter Wert. Andernfalls ist der Standardwert für statische SQL-Anweisungen der Wert STMTNO und für dynamische SQL-Anweisungen der Wert 1.
QUERYTAG	CHAR(20)	Nein	Nein	Identifizierendes Kennzeichen für jede mit EXPLAIN bearbeitet SQL-Anweisung. Für dynamische SQL-Anweisungen, die mit CLP angegeben wurden (außer der EXPLAIN-SQL-Anweisung), ist der Standardwert 'CLP'. Für dynamische SQL-Anweisungen, die mit CLI angegeben wurden (außer der EXPLAIN-SQL-Anweisung), ist der Standardwert 'CLI'. Andernfalls werden Leerzeichen als Standardwert verwendet.
TBNAME	VARCHAR(128)	Ja	Nein	Gibt den Tabellennamen an.
TBCREATOR	VARCHAR(128)	Ja	Nein	Gibt den Namen des Tabellenerstellers an.
PMID	SMALLINT	Ja	Nein	Gibt die Partitionierungszuordnungs-ID an.
TBSPACE	VARCHAR(128)	Ja	Nein	Gibt den Tabellenbereich an, in dem sich die Tabelle befindet.
COLNAMES	CLOB(2M)	Ja	Nein	Gibt die Partitionsspaltennamen durch Kommata getrennt an.
COLCOUNT	SMALLINT	Ja	Nein	Gibt die Anzahl von Partitionierungsspalten an.
REPLICATE	CHAR(1)	Ja	Nein	Gibt an, ob die Partition repliziert wird.
COST	DOUBLE	Ja	Nein	Gibt den Aufwand für die Verwendung der Partition an.
USEIT	CHAR(1)	Ja	Nein	Gibt an, ob die Partition im Modus EVALUATE PARTITION verwendet wird. Eine Partition wird verwendet, wenn USEIT auf 'Y' oder 'y' gesetzt ist.

Tabelle ADVISE_PARTITION

Tabelle 70. Tabelle ADVISE_PARTITION (Forts.). PK bedeutet, dass die Spalte Teil eines Primärschlüssels ist. FK bedeutet, dass die Spalte Teil eines Fremdschlüssels ist.

Spaltenname	Datentyp	Nullwert?	Schlüssel?	Beschreibung
RUN_ID	TIMESTAMP	Ja	FK	Ein Wert, der dem Wert für START_TIME einer Zeile in der Tabelle ADVISE_INSTANCE entspricht, so dass diese Zeile der gleichen Ausführung des Designadvisors zugeordnet wird.

Die Tabelle ADVISE_TABLE

Die Tabelle ADVISE_TABLE speichert die DDL-Anweisungen (Data Definition Language) für die Erstellung von Tabellen unter Verwendung der endgültigen Empfehlungen des Designadvisors für gespeicherte Abfragetabellen (MQTs), MDC-Tabellen und Partitionierung.

Tabelle 71. Tabelle ADVISE_TABLE. PK bedeutet, dass die Spalte Teil eines Primärschlüssels ist. FK bedeutet, dass die Spalte Teil eines Fremdschlüssels ist.

Spaltenname	Datentyp	Nullwert?	Schlüssel?	Beschreibung
RUN_ID	TIMESTAMP	Ja	FK	Ein Wert, der dem Wert für START_TIME einer Zeile in der Tabelle ADVISE_INSTANCE entspricht, so dass diese Zeile der gleichen Ausführung des Designadvisors zugeordnet wird.
TABLE_NAME	VARCHAR(128)	Nein	Nein	Name der Tabelle
TABLE_SCHEMA	VARCHAR(128)	Nein	Nein	Name des Tabellenerstellers
TABLESPACE	VARCHAR(128)	Nein	Nein	Der Tabellenbereich, in dem die Tabelle zu erstellen ist
SELECTION_FLAG	VARCHAR(4)	Nein	Nein	Gibt den Empfehlungstyp an. Gültige Werte sind 'M' für MQT, 'P' für Partitionierung und 'C' für MDC. Dieses Feld kann eine beliebige Untermenge dieser Werte enthalten. Zum Beispiel gibt 'MC' an, dass die Tabelle als MQT und als MDC-Tabelle empfohlen wird.
TABLE_EXISTS	CHAR(1)	Nein	Nein	Auf 'Y' gesetzt, wenn die Tabelle im Datenbankkatalog vorhanden ist.
USE_TABLE	CHAR(1)	Nein	Nein	Auf 'Y' gesetzt, wenn die Tabelle Empfehlungen vom Designadvisor hat.
GEN_COLUMNS	CLOB(2M)	Nein	Nein	Enthält eine Zeichenfolge für generierte Spalten, wenn diese Zeile eine MDC-Empfehlung enthält, die generierte Spalten in der DDL-Anweisung CREATE TABLE fordert.
ORGANIZE_BY	CLOB(2M)	Nein	Nein	Enthält für MDC-Empfehlungen die Klausel ORGANIZE BY für die DDL-Anweisung CREATE TABLE.
CREATION_TEXT	CLOB(2M)	Nein	Nein	Enthält die DDL-Anweisung CREATE TABLE.
ALTER_COMMAND	CLOB(2M)	Nein	Nein	Enthält eine Anweisung ALTER TABLE für die Tabelle.

Die Tabelle ADVISE_WORKLOAD

Die Tabelle ADVISE_WORKLOAD zeigt die Anweisung, die die Auslastung bildet.

Tabelle 72. Die Tabelle ADVISE_WORKLOAD. PK bedeutet, dass die Spalte Teil eines Primärschlüssels ist. FK bedeutet, dass die Spalte Teil eines Fremdschlüssels ist.

Spaltenname	Datentyp	Nullwert?	Schlüssel?	Beschreibung
WORKLOAD_NAME	CHAR(128)	Nein	Nein	Name der Objektgruppe von SQL-Anweisungen (Auslastung), zu der diese Anweisung gehört.
STATEMENT_NO	INTEGER	Nein	Nein	Anweisungsnummer in der Auslastung, auf die sich diese EXPLAIN-Informationen beziehen.
STATEMENT_TEXT	CLOB(1M)	Nein	Nein	Inhalt der SQL-Anweisung
STATEMENT_TAG	VARCHAR(256)	Nein	Nein	Identifizierendes Kennzeichen für jede mit EXPLAIN bearbeitete SQL-Anweisung
FREQUENCY	INTEGER	Nein	Nein	Gibt an, wie oft diese Anweisung in der Auslastung erscheint.
IMPORTANCE	DOUBLE	Nein	Nein	Wichtigkeit der Anweisung
WEIGHT	DOUBLE	Nein	Nein	Priorität der Anweisung
COST_BEFORE	DOUBLE	Ja	Nein	Der Aufwand (in Timerons) der Abfrage, wenn die empfohlenen Indizes nicht erstellt werden.
COST_AFTER	DOUBLE	Ja	Nein	Der Aufwand (in Timerons) der Abfrage, wenn die empfohlenen Indizes erstellt werden.
COMPILABLE	CHAR(17)	Ja	Nein	Gibt Abfragekompilierfehler an, die beim Versuch, die Anweisung vorzubereiten, aufgetreten sind. Wenn diese Spalte den Wert NULL hat oder der Wert nicht mit SQLCA beginnt, konnte die SQL-Abfrage durch 'db2advis' kompiliert werden. Wenn ein Kompilierfehler von 'db2advis' oder dem Designadvisor gefunden wird, besteht der Wert der Spalte COMPILABLE aus einem acht Zeichen langen SQLCA.sqlcaid-Feld, gefolgt von einem Doppelpunkt (:) sowie einem acht Zeichen langen SQLCA.sqlstate-Feld, das den Rückkehrcode für die SQL-Anweisung darstellt.

Anhang C. SQL-EXPLAIN-Tools

Sie können die EXPLAIN-Tools db2expln und dynexpln verwenden, um sich einen Einblick in den für eine bestimmte SQL-Anweisung ausgewählten Plan zu verschaffen. Sie können auch die integrierte EXPLAIN-Einrichtung in der Steuerzentrale zusammen mit Visual Explain verwenden, wenn Sie eine Beschreibung für den für eine bestimmte SQL-Anweisung ausgewählten Zugriffsplan erhalten möchten. Mit der EXPLAIN-Einrichtung können sowohl dynamische als auch statische SQL-Anweisungen bearbeitet werden. Ein Unterschied zu den EXPLAIN-Tools besteht darin, dass mit Visual Explain die EXPLAIN-Informationen in einem grafischen Format dargestellt werden. Ansonsten ist die Detaillierungsebene beider Methoden gleichwertig.

Damit Sie die Ausgabe von `db2expln` und `dynexpln` optimal verwenden können, müssen Sie mit folgenden Punkten vertraut sein:

- Die verschiedenen unterstützten SQL-Anweisungen und die für diese Anweisungen verwendete Terminologie (z. B. Vergleichselemente in einer Anweisung SELECT)
- Der Zweck eines Pakets (Zugriffsplans)
- Der Zweck und Inhalt der Systemkatalogtabellen
- Allgemeine Konzepte zur Anwendungsoptimierung

Die Themen dieses Abschnitts enthalten Informationen zu `db2expln` und `dynexpln`.

SQL-EXPLAIN-Tools

Das Tool `db2expln` beschreibt den Zugriffsplan, der für SQL-Anweisungen ausgewählt wird. Mit diesem Tool kann eine schnelle Erläuterung des ausgewählten Zugriffsplans abgerufen werden, wenn keine EXPLAIN-Daten erfasst wurden. Für statisches SQL untersucht `db2expln` die Pakete, die in den Systemkatalogtabellen gespeichert sind. Für dynamisches SQL untersucht `db2expln` die Abschnitte im SQL-Cache.

Das Tool `dynexpln` kann ebenfalls zur Beschreibung des für dynamische Anweisungen ausgewählten Zugriffsplans verwendet werden. Es erstellt ein statisches Paket für die Anweisungen und verwendet dann das Tool `db2expln`, um sie zu beschreiben. Da dynamisches SQL jedoch auch mit `db2expln` untersucht werden kann, wird dieses Dienstprogramm nur aus Gründen der Abwärtskompatibilität weiterhin bereitgestellt.

Die EXPLAIN-Tools (`db2expln` und `dynexpln`) befinden sich im Unterverzeichnis `bin` im Verzeichnis `sqllib` Ihres Exemplars. Wenn sich `db2expln` und `dynexpln` nicht in Ihrem aktuellen Verzeichnis befinden, müssen sie sich in einem Verzeichnis befinden, das in Ihrer Umgebungsvariablen PATH definiert ist.

Das Programm `db2expln` stellt eine Verbindung zu einer Datenbank her und bindet sich mit Hilfe der Dateien `db2expln.bnd`, `db2exsrv.bnd` und `db2exdyn.bnd` selbst an diese Datenbank, wenn auf die Datenbank zum ersten Mal zugegriffen wird.

Zum Ausführen von `db2expln` müssen Sie über das Zugriffsrecht SELECT für die Systemkatalogsichten sowie über das Zugriffsrecht EXECUTE für die Pakete

db2expln, db2exsrv und db2exdyn verfügen. Zum Ausführen von dynexpln müssen Sie die Berechtigung BINDADD für die Datenbank besitzen, und das Schema, das Sie zur Verbindung zur Datenbank verwenden, muss vorhanden sein oder Sie müssen die Berechtigung IMPLICIT_SCHEMA für die Datenbank haben. Zur EXPLAIN-Bearbeitung dynamischer SQL-Anweisungen mit db2expln oder dynexpln müssen Sie außerdem alle Zugriffsrechte haben, die für die mit EXPLAIN bearbeitete SQL-Anweisung erforderlich sind. (Wenn Sie über SYSADM- oder DBADM-Berechtigung verfügen, haben Sie automatisch alle erforderlichen Berechtigungsstufen.)

Zugehörige Konzepte:

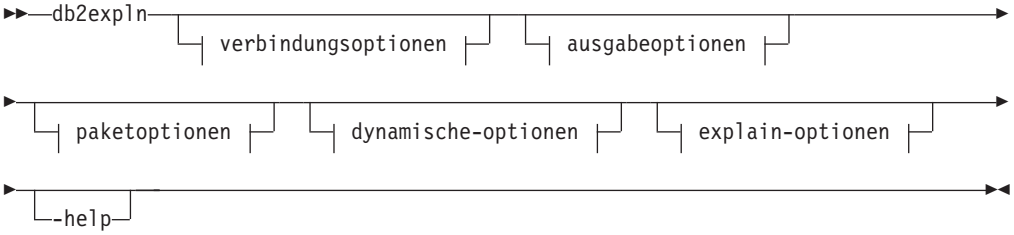
- „SQL-EXPLAIN-Einrichtung" auf Seite 223
- „Richtlinien zur Erfassung von EXPLAIN-Informationen" auf Seite 234
- „Richtlinien zur Analyse von EXPLAIN-Informationen" auf Seite 236

db2expln

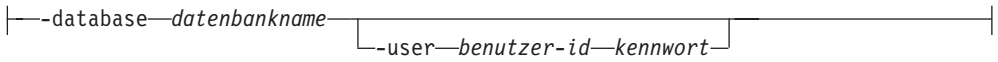
Die folgenden Abschnitte beschreiben die Syntax und die Parameter für *db2expln* und geben Hinweise zur Verwendung.

db2expln - SQL-EXPLAIN-Tool

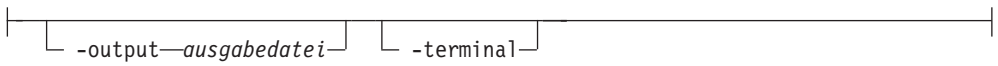
Befehlssyntax:



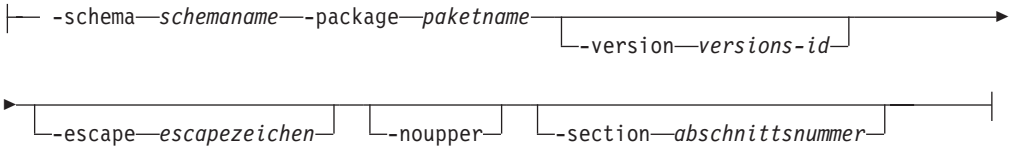
verbindungsoptionen:



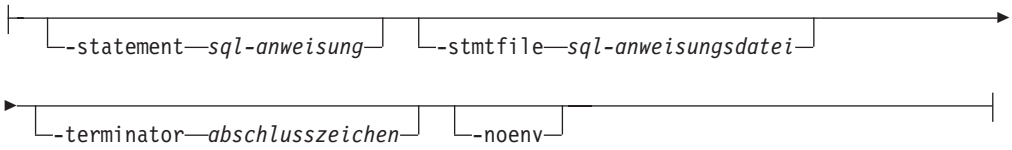
ausgabeoptionen:



paketoptionen:



dynamische-optionen:



explain-optionen:

Befehlsparameter:
Die Optionen können in beliebiger Reihenfolge angegeben werden.

verbindungsoptionen:

Diese Optionen geben die Datenbank an, zu der eine Verbindung herzustellen ist, sowie alle für die Verbindung erforderlichen Optionen. Die Verbindungsoptionen sind außer bei Angabe der Option **-help** immer erforderlich.

-database *datenbankname*
Der Name der Datenbank, die die mit EXPLAIN zu bearbeitenden Pakete enthält.

Aus Gründen der Abwärtskompatibilität können Sie **-d** anstelle von **-database** verwenden.

-user *benutzer-id kennwort*
Die Berechtigungs-ID und das Kennwort zur Einrichtung der Datenbankverbindung. Sowohl *benutzer-id* als auch *kennwort* müssen gemäß den DB2®-Namenskonventionen gültig sein und von der Datenbank erkannt werden.

Aus Gründen der Abwärtskompatibilität können Sie **-u** anstelle von **-user** verwenden.

ausgabeoptionen:

Diese Optionen geben an, wohin die Ausgabe von `db2expln` geleitet werden soll. Sofern nicht die Option **-help** angegeben wird, müssen Sie mindestens eine Ausgabeoption angeben. Wenn Sie beide Optionen angeben, wird die Ausgabe in eine Datei und an das Terminal geleitet.

-output *ausgabedatei*
Die Ausgabe von `db2expln` wird in die von Ihnen angegebene Datei geschrieben.

Aus Gründen der Abwärtskompatibilität können Sie **-o** anstelle von **-output** verwenden.

-terminal
Die Ausgabe von `db2expln` wird an das Terminal geleitet.

Aus Gründen der Abwärtskompatibilität können Sie **-t** anstelle von **-terminal** verwenden.

paketoptionen:

Diese Optionen geben ein oder mehrere Pakete und Abschnitte (Sections) an, die mit EXPLAIN zu bearbeiten sind. Nur statisches SQL in den Paketen und Abschnitten wird mit EXPLAIN bearbeitet.

Anmerkung: Wie bei einem LIKE-Vergleichselement können Sie Platzhalterzeichen, d. h. das Prozentzeichen (%) und das Unterstreichungszeichen (_), verwenden, um den *schemanamen*, den *paketnamen* und die *versions-id* anzugeben.

-schema *schemaname*
Das Schema des Pakets oder der Pakete, die mit EXPLAIN zu bearbeiten sind. Aus Gründen der Abwärtskompatibilität können Sie **-c** anstelle von **-schema** verwenden.

-package *paketname*
Der Name des Pakets oder der Pakete, die mit EXPLAIN zu bearbeiten sind. Aus Gründen der Abwärtskompatibilität können Sie **-p** anstelle von **-package** verwenden.

-version *versions-id*
Die Versionskennung (ID) des Pakets oder der Pakete, die mit EXPLAIN zu bearbeiten sind. Der Standardwert für die Version ist eine leere Zeichenfolge.

-escape *escapezeichen*
Das Zeichen, das als Escapezeichen beim Musterabgleich in den Angaben *schemaname*, *paketname* und *versions-id* zu verwenden ist.

Zum Beispiel sieht der `db2expln`-Befehl zur Bearbeitung des Pakets TESTID.CALC% wie folgt aus:

```
db2expln -schema TESTID -package CALC% ....
```

Mit diesem Befehl würden jedoch auch alle anderen Pläne, die mit CALC beginnen, von EXPLAIN bearbeitet. Wenn jedoch nur das Paket TESTID.CALC% bearbeitet werden soll, muss das Escapezeichen verwendet werden. Wenn Sie das Ausrufungszeichen (!) als Escapezeichen angeben, können Sie den Befehl in folgende Form umwandeln: `db2expln -schema TESTID -escape ! -package CALC!% ...` . Nun wird das Ausrufungszeichen (!) als Escapezeichen verwendet und daher die Zeichenfolge !% als das Prozentzeichen (%) interpretiert und nicht als Platzhalterzeichen für eine beliebige Zeichenfolge. Es gibt kein Standardescapezeichen.

Aus Gründen der Abwärtskompatibilität können Sie **-e** anstelle von **-escape** verwenden.

Anmerkung: Geben Sie zur Vermeidung von Problemen nicht das Escapezeichen des Betriebssystems als Escapezeichen für `db2expln` an.

-noupper
Gibt an, dass *schemaname*, *paketname* und *versions-id* nicht in Großbuchstaben umzusetzen sind, bevor nach entsprechenden Paketen gesucht wird. Standardmäßig werden die Werte dieser Variablen in Großbuchstaben umgesetzt, bevor nach Paketen gesucht wird. Diese Option ermöglicht eine Verwendung der Werte in exakt der eingegebenen Form.

Aus Gründen der Abwärtskompatibilität können Sie **-l** (kleines L, nicht die Ziffer 1) anstelle von **-noupper** verwenden.

-section *abschnittsnummer*
Die Nummer des mit EXPLAIN zu bearbeitenden Abschnitts im ausgewählten Paket bzw. in den ausgewählten Paketen.

Zur Bearbeitung aller Abschnitte in jedem Paket geben Sie die Ziffer 0 an. Dies ist der Standardwert. Wenn Sie diese Option nicht angeben oder wenn *schemaname*, *paketname* oder *versions-id* ein Platzhalterzeichen enthalten, werden alle Abschnitte angezeigt.

Zur Ermittlung von Abschnittsnummern fragen Sie die Systemkatalogsicht SYSCAT.STATEMENTS ab. Im Handbuch *SQL Reference* finden Sie eine Beschreibung der Systemkatalogsichten.

Aus Gründen der Abwärtskompatibilität können Sie **-s** anstelle von **-section** verwenden.

dynamische-optionen:

Diese Optionen geben eine oder mehrere dynamische SQL-Anweisungen zur Bearbeitung durch EXPLAIN an.

-statement *sql-anweisung*
Die SQL-Anweisung, die dynamisch vorzubereiten und mit EXPLAIN zu bearbeiten ist. Zur Bearbeitung mehrerer Anweisungen geben Sie entweder mit der Option **-stmtfile** eine Datei an, die die zu bearbeitenden SQL-Anweisungen enthält, oder definieren mit der Option **-terminator** ein Abschlusszeichen, das zur Trennung von Anweisungen in der Option **-statement** verwendet werden kann.

Aus Gründen der Kompatibilität mit `dynexpln` können Sie **-q** anstelle von **-statement** verwenden.

-stmtfile *sql-anweisungsdatei*
Eine Datei, die eine oder mehrere SQL-Anweisungen enthält, die dynamisch vorzubereiten und mit EXPLAIN zu bearbeiten sind. Standardmäßig wird angenommen, dass jede Zeile der Datei eine eigene SQL-Anweisung darstellt. Wenn sich Anweisungen über mehrere Zeilen erstrecken müssen, verwenden Sie die Option **-terminator**, um das Zeichen anzugeben, mit dem das Ende einer SQL-Anweisung markiert wird.

Aus Gründen der Kompatibilität mit `dynexpln` können Sie **-f** anstelle von **-stmtfile** verwenden.

-terminator *abschlusszeichen*
Das Zeichen, mit dem das Ende dynamischer SQL-Anweisungen angegeben wird. Standardmäßig gibt die Option **-statement** eine einzelne SQL-Anweisung an, und jede Zeile in der mit der Option **-stmtfile** angegebenen Datei wird als getrennte SQL-Anweisung behandelt. Das von Ihnen angegebene Abschlusszeichen kann dazu verwendet werden, mehrere SQL-Anweisungen über die Option **-statement** anzugeben oder Anweisungen in der mit der Option **-stmtfile** angegebenen Datei über mehrere Zeilen zu verteilen.

Aus Gründen der Kompatibilität mit `dynexpln` können Sie **-z** anstelle von **-terminator** verwenden.

-noenv

Gibt an, dass dynamische Anweisungen, die die Kompilierumgebung ändern, nach der Bearbeitung durch EXPLAIN nicht ausgeführt werden sollen.

Standardmäßig führt db2expln jede der folgenden Anweisungen nach der Bearbeitung durch EXPLAIN aus:

```
SET CURRENT DEFAULT TRANSFORM GROUP
SET CURRENT DEGREE
SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION
SET CURRENT QUERY OPTIMIZATION
SET CURRENT REFRESH AGE
SET PATH
SET SCHEMA
```

Diese Anweisungen ermöglichen es, den ausgewählten Plan für nachfolgende dynamische SQL-Anweisungen zu ändern, die von db2expln verarbeitet werden.

Wenn Sie **-noenv** angeben, werden diese Anweisungen mit EXPLAIN bearbeitet, jedoch nicht ausgeführt.

Es ist erforderlich, entweder **-statement** oder **-stmtfile** anzugeben, um dynamisches SQL mit EXPLAIN zu bearbeiten. Beide Optionen können auch in einem Aufruf von db2expln zusammen angegeben werden.

explain-optionen:

Diese Optionen legen fest, welche zusätzlichen Informationen in den EXPLAIN-Plänen bereitgestellt werden.

-graph Zeigt die Plandiagramme des Optimierungsprogramms. Jeder Abschnitt wird untersucht, und das ursprüngliche Plandiagramm des Optimierungsprogramms wird, wie von Visual Explain dargestellt, konstruiert. Beachten Sie, dass das generierte Diagramm eventuell nicht exakt mit dem Diagramm von Visual Explain übereinstimmt.

Aus Gründen der Abwärtskompatibilität können Sie **-g** anstelle von **-graph** angeben.

-opids Zeigt die Operator-IDs in dem mit EXPLAIN bearbeiteten Plan.

Die Operator-IDs ermöglichen das Abgleichen der Ausgabe von db2expln mit der Ausgabe der EXPLAIN-Einrichtung. Beachten Sie, dass nicht alle Operatoren eine ID-Nummer haben und dass einige ID-Nummern, die in der Ausgabe der EXPLAIN-Einrichtung vorkommen, nicht in der Ausgabe von db2expln enthalten sind.

Aus Gründen der Abwärtskompatibilität können Sie **-i** anstelle von **-opids** angeben.

-help Zeigt den Hilfetext zu db2expln an. Wenn diese Option angegeben wird, findet keine Bearbeitung von Paketen durch EXPLAIN statt.

Der größte Teil der Befehlszeile wird in der gespeicherten Prozedur *db2exsrv* ausgeführt. Um Hilfe für alle verfügbaren Optionen zu erhalten, ist es erforderlich, **verbindungsoptionen** zusammen mit der Option **-help** anzugeben. Geben Sie zum Beispiel Folgendes ein:

```
db2expln -help -database SAMPLE
```

Aus Gründen der Abwärtskompatibilität können Sie auch **-h** oder **-?** angeben.

Hinweise zur Verwendung:

Wenn Sie nicht die Option **-help** angeben, müssen Sie entweder Paketoptionen oder dynamische Optionen angeben. Sie können in einem Aufruf von `db2expln` sowohl Pakete als auch dynamisches SQL bearbeiten.

Einige der oben erläuterten Optionsangaben haben für Ihr Betriebssystem vielleicht eine spezielle Bedeutung, so dass sie möglicherweise in der Befehlszeile für den Befehl `db2expln` nicht korrekt interpretiert werden. Sie können diese Zeichen möglicherweise jedoch eingeben, indem Sie ihnen ein Escapezeichen des Betriebssystems voranstellen. Weitere Informationen zu dieser Möglichkeit finden Sie in der Dokumentation zu Ihrem Betriebssystem. Stellen Sie sicher, dass Sie nicht versehentlich das Escapezeichen des Betriebssystems als Escapezeichen für `db2expln` angeben.

Hilfenachrichten und Nachrichten zum Anfangsstatus, die von `db2expln` erstellt werden, werden an die Standardausgabeeinheit geleitet. Alle Dialognachrichten und anderen Statusnachrichten, die durch das EXPLAIN-Tool generiert werden, werden an die Standardausgabeeinheit für Fehler geleitet. EXPLAIN-Text wird je nach der gewählten Ausgabeoption an die Standardausgabeeinheit oder in eine Datei geleitet.

Beispiele:

Zur Bearbeitung mehrerer Pläne in einem Aufruf von `db2expln` verwenden Sie die Optionen **-package**, **-schema** und **-version** und geben Zeichenfolgekonstanten für Pakete und Ersteller mit LIKE-Mustern an. Das Unterstreichungszeichen (_) kann also als Platzhalterzeichen für ein Zeichen und das Prozentzeichen (%) als Platzhalterzeichen für null oder mehr Zeichen verwendet werden.

Wenn alle Abschnitte für alle Pakete in der Datenbank SAMPLE bearbeitet und die Ergebnisse in der Datei **my.exp** gespeichert werden sollen, geben Sie Folgendes ein:

```
db2expln -database SAMPLE -schema % -package %  -output my.exp
```

Nehmen Sie als weiteres Beispiel an, dass ein Benutzer eine CLP-Prozedurdatei mit dem Namen ″statements.db2″ hat und die Anweisungen in der Datei mit EXPLAIN bearbeiten will. Die Datei enthält die folgenden Anweisungen:

```
SET PATH=SYSIBM, SYSFUN, DEPT01, DEPT93@
SELECT EMPNO, TITLE(JOBID) FROM EMPLOYEE@
```

Zur Bearbeitung dieser Anweisungen geben Sie den folgenden Befehl ein:

```
db2expln -database DEPTDATA -stmtfile statements.db2 -terminator @ -terminal
```

Zugehörige Konzepte:

- „SQL-EXPLAIN-Tools” auf Seite 637
- „Beschreibung der Ausgabe von db2expln und dynexpln” auf Seite 645
- „Beispiele für die Ausgabe von db2expln und dynexpln” auf Seite 665

Hinweise zur Verwendung von db2expln

Es folgen einige allgemeine Nachrichten, die db2expln ausgeben kann:

- `No packages found for database package pattern: "<creator>".<package> with version "<version>"`

  Diese Nachricht wird in der Ausgabe angezeigt, wenn keine Pakete in der Datenbank gefunden wurden, die dem angegebenen Muster entsprachen.

- `Bind messages can be found in db2expln.msg`

  Diese Nachricht wird in der Ausgabe angezeigt, wenn das Binden von db2expln.bnd nicht erfolgreich war. Weitere Informationen über die aufgetretenen Fehler finden Sie in der Datei db2expln.msg im aktuellen Verzeichnis.

- `Section number overridden to 0 (all sections) for potential multiple packages.`

  Die Nachricht, dass die Abschnittsnummer mit 0 überschrieben wurde, wird in der Ausgabe angezeigt, wenn von db2expln mehrere Pakete bearbeitet werden können. Diese Aktion wird ausgeführt, wenn eines der Platzhalterzeichen in den Eingabeargumenten für Pakete oder Ersteller verwendet wird.

- `Bind messages for <bind file> can be found in <message file>`

  Diese Nachricht wird angezeigt, wenn das Binden der angegebenen Bindedatei nicht erfolgreich war. Weitere Informationen über die aufgetretenen Fehler finden Sie in der angegebenen Nachrichtendatei auf dem Datenbankserver.

- `No static sections qualify from package.`

  Diese Nachricht wird in der Ausgabe angezeigt, wenn das angegebene Paket nur dynamische SQL-Anweisungen enthält, was bedeutet, dass keine statischen Abschnitte existieren.

- `Package "<creator>"."<package>", "<version>", is not valid. Rebind the package and then rerun db2expln.`

  Die Nachricht wird in der Ausgabe angezeigt, wenn das angegebene Paket momentan ungültig ist. Führen Sie, wie in der Nachricht angegeben, den Befehl BIND oder REBIND für den Plan neu aus, um erneut ein gültiges Paket in der Datenbank zu erstellen, und wiederholen Sie anschließend db2expln.

Ausgeschlossene SQL-Anweisungen: Die folgenden Anweisungen werden von EXPLAIN nicht bearbeitet:

- BEGIN/END DECLARE SECTION
- BEGIN/END COMPOUND
- INCLUDE
- WHENEVER
- COMMIT und ROLLBACK
- CONNECT
- OPEN Cursor
- FETCH
- CLOSE Cursor
- PREPARE
- EXECUTE
- EXECUTE IMMEDIATE
- DESCRIBE
- Dynamisches DECLARE CURSOR
- SQL-Steueranweisungen

Jede Unteranweisung innerhalb einer Compound-SQL-Anweisung kann über einen eigenen Abschnitt verfügen, der von **db2expln** bearbeitet werden kann.

dynexpln

Das Tool dynexpln ist aus Gründen der Abwärtskompatibilität weiterhin verfügbar. Sie können jedoch die **dynamischen-optionen** von db2expln verwenden, um alle Funktionen von dynexpln auszuführen.

Wenn Sie die dynamischen Optionen von db2expln verwenden, wird die Anweisung wie richtiges dynamisches SQL vorbereitet (PREPARE) und der generierte Plan aus dem SQL-Cache heraus mit EXPLAIN bearbeitet. Diese EXPLAIN-Ausgabemethode bietet genauere Zugriffspläne als das Tool dynexpln, das die Anweisung als statisches SQL vorbereitet. Sie ermöglicht darüber hinaus die Verwendung von Einrichtungen, die nur in dynamischem SQL verfügbar sind, wie zum Beispiel Parametermarken.

Zugehörige Konzepte:

- „SQL-EXPLAIN-Tools" auf Seite 637
- „Beispiele für die Ausgabe von db2expln und dynexpln" auf Seite 665

Informationen der EXPLAIN-Ausgabe

Die folgenden Abschnitte beschreiben die Art der Informationen, die von *db2expln* und *dynexpln* bereitgestellt werden können.

Beschreibung der Ausgabe von db2expln und dynexpln

In der Ausgabe werden die EXPLAIN-Informationen für jedes Paket in den beiden folgenden Teilen dargestellt:

- Paketinformationen wie das Datum des Bindevorgangs und relevante Bindeoptionen
- Abschnittsinformationen wie die Abschnittsnummer, gefolgt von der mit EXPLAIN bearbeiteten SQL-Anweisung. Unter den Abschnittsinformationen befindet sich die EXPLAIN-Ausgabe des für die SQL-Anweisung ausgewählten Zugriffsplans.

Die Schritte eines Zugriffsplans oder Abschnitts werden in der Reihenfolge aufgeführt, in der sie vom Datenbankmanager ausgeführt werden. Jeder Hauptschritt wird als linksbündig ausgerichtete Überschrift angezeigt. Informationen zum Schritt werden darunter eingerückt angezeigt. Am linken Rand der EXPLAIN-Ausgabe für den Zugriffsplan befinden sich Einrückungsbalken. Diese Balken markieren gleichzeitig den Geltungsbereich der Operation. Operationen niedrigerer Einrückungsstufe in derselben Operation, das heißt solche, die weiter rechts angezeigt werden, werden verarbeitet, bevor zur vorherigen Einrückungsstufe zurückgekehrt wird.

Beachten Sie, dass der ausgewählte Zugriffsplan auf einer erweiterten Version der ursprünglichen SQL-Anweisung basiert, die in der Ausgabe gezeigt wird. Beispielsweise kann die ursprüngliche Anweisung die Aktivierung von Auslösern und Integritätsbedingungen bewirken. Darüber hinaus kann die Komponente des SQL-Compilers zum Umschreiben von Abfragen die SQL-Anweisung in ein äquivalentes, jedoch effizienteres Format umschreiben. Alle diese Faktoren sind in den Informationen enthalten, die das Optimierungsprogramm verwendet, wenn es den effizientesten Plan zum Ausführen der Anweisung ermittelt. Daher kann sich der in der EXPLAIN-Ausgabe angezeigte Zugriffsplan erheblich von dem unterscheiden, der für die ursprüngliche SQL-Anweisung vermutet werden könnte. Die SQL-

EXPLAIN-Einrichtung, zu der die EXPLAIN-Tabellen, der Modus SET CURRENT EXPLAIN und Visual Explain gehören, zeigt die tatsächlich für die Optimierung verwendete SQL-Anweisung in Form einer SQL-ähnlichen Anweisung, die durch Rückübersetzung der internen Darstellung der Abfrage erstellt wird.

Beim Vergleichen der Ausgabe von `db2expln` oder `dynexpln` mit der Ausgabe der EXPLAIN-Einrichtung kann die Option für die Operator-ID (**-opids**) sehr nützlich sein. Jedesmal, wenn `db2expln` bzw. `dynexpln` die Verarbeitung eines neuen Operators aus der EXPLAIN-Einrichtung beginnt, wird die Operator-ID links neben dem von EXPLAIN gezeigten Plan ausgegeben. Die Operator-ID kann zum Abgleichen der Schritte in den verschiedenen Darstellungen des Zugriffsplans verwendet werden. Beachten Sie, dass es nicht immer eine Eins-zu-eins-Entsprechung gibt zwischen den Operatoren in der Ausgabe der EXPLAIN-Einrichtung und den Operationen, die in der Ausgabe von `db2expln` und `dynexpln` angezeigt werden.

Zugehörige Konzepte:

- „dynexpln" auf Seite 645
- „Tabellenzugriffsinformationen" auf Seite 646
- „Informationen zu temporären Tabellen" auf Seite 652
- „Verknüpfungsinformationen" auf Seite 654
- „Datenstrominformationen" auf Seite 656
- „Informationen zu INSERT, UPDATE und DELETE" auf Seite 657
- „Informationen zur Vorbereitung von Block- und Zeilen-IDs" auf Seite 657
- „Informationen zu Spaltenberechnungen (Aggregation)" auf Seite 658
- „Informationen zur Parallelverarbeitung" auf Seite 659
- „Informationen zu Abfragen auf zusammengeschlossene Datenbanken" auf Seite 662
- „Verschiedene Informationen" auf Seite 663

Zugehörige Referenzen:

- „db2expln - SQL-EXPLAIN-Tool" auf Seite 638

Tabellenzugriffsinformationen

Diese Angabe zeigt den Namen und den Typ der Tabelle an, auf die zugegriffen wird. Es werden zwei Formate verwendet:

1. Drei Typen von regulären Tabellen:
   - Zugriff auf Tabellenname (Access Table Name):

     ```
     Access Table Name = schema.name  ID = ts,n
     ```

     Dabei gilt Folgendes:
     - *schema.name* ist der vollständig qualifizierte Name der Tabelle, auf die zugegriffen wird.
     - *ID* ist die entsprechende TABLESPACEID und TABLEID aus dem Katalog SYSCAT.TABLES für die Tabelle.
   - Zugriff auf Hierarchietabellenname (Access Hierarchy Table Name):

     ```
     Access Hierarchy Table Name = schema.name  ID = ts,n
     ```

     Dabei gilt Folgendes:
     - *schema.name* ist der vollständig qualifizierte Name der Tabelle, auf die zugegriffen wird.

  - *ID* ist die entsprechende TABLESPACEID und TABLEID aus dem Katalog SYSCAT.TABLES für die Tabelle.
- Zugriff auf den Namen einer gespeicherten Abfragetabelle (Access Materialized Query Table Name):

  ```
  Access Materialized Query Table Name = schema.name  ID = ts,n
  ```

  Dabei gilt Folgendes:
  - *schema.name* ist der vollständig qualifizierte Name der Tabelle, auf die zugegriffen wird.
  - *ID* ist die entsprechende TABLESPACEID und TABLEID aus dem Katalog SYSCAT.TABLES für die Tabelle.

2. Zwei Typen von temporären Tabellen:
   - Zugriff auf ID für temporäre Tabellen (Access Temporary Table ID):

     ```
     Access Temp Table  ID = tn
     ```

     Dabei gilt Folgendes:
     - *ID* ist die entsprechende Kennung, die von `db2expln` zugeordnet wurde.
   - Zugriff auf ID für deklarierte temporäre Tabellen (Access Declared Global Temporary Table ID):

     ```
     Access Global Temp Table  ID = ts,tn
     ```

     Dabei gilt Folgendes:
     - *ID* ist die entsprechende TABLESPACEID aus dem Katalog SYSCAT.TABLES für die Tabelle (`ts`) und die entsprechende von `db2expln` zugeordnete Kennung (`tn`)

Nach der Angabe über den Zugriff auf die Tabelle folgen weitere Angaben, die den Zugriff weiter beschreiben. Diese Angaben werden unter der Angabe über den Zugriff auf die Tabelle eingerückt. Folgende Angaben sind möglich:
- Anzahl von Spalten
- Blockzugriff
- Parallelsuche
- Suchrichtung
- Zeilenzugriffsmethode
- Sperrabsichten
- Vergleichselemente
- Verschiedene Angaben

Anzahl von Spalten

Die folgende Angabe zeigt die Anzahl von Spalten an, die aus jeder Zeile der Tabelle verwendet werden:

```
#Columns = n
```

Blockzugriff

Die folgende Angabe zeigt an, dass für die Tabelle ein oder mehrere Dimensionsblockindizes definiert sind:

```
Clustered by Dimension for Block Index Access
```

Wenn dieser Text nicht angezeigt wird, wurde die Tabelle ohne die Klausel DIMENSION erstellt.

Parallelsuche

Die folgende Angabe zeigt an, dass der Datenbankmanager mehrere Subagenten zum parallelen Lesen aus der Tabelle verwendet:

```
Parallel Scan
```

Wenn diese Angabe fehlt, wird die Tabelle nur von einem Agenten (bzw. Subagenten) gelesen.

Suchrichtung

Die folgende Angabe zeigt an, dass der Datenbankmanager Zeilen in umgekehrter Reihenfolge liest:

```
Scan Direction = Reverse
```

Wenn dieser Text nicht angezeigt wird, wird in Vorwärtsrichtung gesucht (dies ist der Standardwert).

Zeilenzugriffsmethode

Eine der folgenden Angaben zur Art und Weise des Zugriffs auf die Zeilen in der Tabelle, die den angegebenen Bedingungen entsprechen, wird angezeigt:

- Die Angabe `Relation Scan` bedeutet, dass die Tabelle sequenziell durchsucht wird, um die den angegebenen Bedingungen entsprechenden Zeilen zu ermitteln.
  - Die folgende Angabe bedeutet, dass kein Vorablesezugriff auf Daten erfolgt:

    ```
    Relation Scan
    |  Prefetch: None
    ```

  - Die folgende Angabe bedeutet, dass das Optimierungsprogramm die Anzahl der Seiten, die vorabgelesen werden, vorbestimmt hat:

    ```
    Relation Scan
    |  Prefetch: n Pages
    ```

  - Die folgende Angabe weist darauf hin, dass die Daten wahrscheinlich vorabgelesen werden:

    ```
    Relation Scan
    |  Prefetch: Eligible
    ```

  - Die folgende Angabe bedeutet, dass die Identifizierung der Zeilen, die den angegebenen Kriterien entsprechen, und der Zugriff auf sie über einen Index erfolgen:

    ```
    Index Scan:  Name = schema.name  ID = xx
    | Index typ
    | Index Columns:
    ```

    Dabei gilt Folgendes:

    - *schema.name* ist der vollständig qualifizierte Name des Index, der durchsucht wird.
    - *ID* ist die entsprechende Spalte IID in der Katalogsicht SYSCAT.INDEXES.
    - Der Indextyp ist einer der folgenden:

      ```
      Regular Index (Not Clustered)
      Regular Index (Clustered)
      Dimension Block Index
      Composite Dimension Block Index
      ```

      Diesen Angaben folgen jeweils eine Zeile für jede Spalte im Index. Jede Spalte im Index wird in einer der folgenden Formen aufgeführt:

```
n: spaltenname (Ascending)
n: spaltenname (Descending)
n: spaltenname (Include Column)
```

Die folgenden Angaben präzisieren die Art der Indexsuche:

- Die bereichsbegrenzenden Vergleichselemente für den Index werden folgendermaßen angegeben:

  ```
  #Key Columns = n
  |  Start Key: xxxxx
  |  Stop Key: xxxxx
  ```

  Dabei ist xxxxx eine der folgenden Angaben:

  - `Start of Index`
  - `End of Index`
  - `Inclusive Value:` oder `Exclusive Value:`

    Ein inklusiver Schlüsselwert wird in die Indexsuche mit einbezogen. Ein exklusiver Schlüsselwert wird in der Suchoperation nicht berücksichtigt. Der Wert für den Schlüssel wird durch eine der folgenden Zeilen für jeden Teil des Schlüssels angegeben:

    ```
    n: 'zeichenfolge'
    n: nnn
    n: jjjj-mm-tt
    n: ss:mm:ss
    n: jjjj-mm-tt ss:mm:ss.uuuuuu
    n: NULL
    n: ?
    ```

    Wenn eine Literalzeichenfolge angezeigt wird, werden nur die ersten 20 Zeichen angegeben. Ist die Zeichenfolge länger als 20 Zeichen, wird dieses durch `...` am Ende der Zeichenfolge angezeigt. Einige Schlüssel können erst bestimmt werden, wenn der Abschnitt (section) ausgeführt wird. Dies wird durch den Wert `?` angezeigt.

- `Index-Only Access`

  Wenn alle benötigten Spalten aus dem Indexschlüssel abgerufen werden können, wird diese Angabe für reinen Indexzugriff angezeigt, und es wird nicht auf Tabellendaten zugegriffen.

- Die folgende Angabe bedeutet, dass kein Vorablesezugriff auf die Indexseiten erfolgt:

  ```
  Index Prefetch: None
  ```

- Die folgende Angabe bedeutet, dass Indexseiten wahrscheinlich vorabgelesen werden:

  ```
  Index Prefetch: Eligible
  ```

- Die folgende Angabe bedeutet, dass kein Vorablesezugriff auf Datenseiten erfolgt:

  ```
  Data Prefetch: None
  ```

- Die folgende Angabe weist darauf hin, dass Datenseiten wahrscheinlich vorabgelesen werden:

  ```
  Data Prefetch: Eligible
  ```

- Wenn Vergleichselemente existieren, die an den Indexmanager übermittelt werden können, um bei der Qualifizierung von Indexeinträgen zu helfen, zeigt die folgende Angabe die Anzahl der Vergleichselemente:

  ```
  Sargable Index Predicate(s)
  |  #Predicates = n
  ```

– Wenn auf die den Bedingung entsprechenden Zeilen über die Zeilen-IDs (RIDs) zugegriffen wird, die zuvor im Zugriffsplan vorbereitet wurden, wird dies durch die folgende Angabe gezeigt:

```
Fetch Direct Using Row IDs
```

Wenn für die Tabelle ein oder mehrere Blockindizes definiert sind, kann der Zugriff auf die Zeilen entweder über Block-IDs oder über Zeilen-IDs erfolgen. Dies wird wie folgt angegeben:

```
Fetch Direct Using Block or Row IOs
```

Sperrabsicht

Für jeden Tabellenzugriff wird die Art der Sperre, die auf Tabellen- und Zeilenebene aktiviert werden soll, mit der folgenden Angabe angezeigt:

```
Lock Intents
|  Table: xxxx
|  Row  : xxxx
```

Folgende Werte sind für eine Tabellensperre möglich:
- Exclusive
- Intent Exclusive
- Intent None
- Intent Share
- Share
- Share Intent Exclusive
- Super Exclusive
- Update

Folgende Werte sind für eine Zeilensperre möglich:
- Exclusive
- Next Key Exclusive (wird nicht in der Ausgabe von `db2expln` angezeigt)
- None
- Share
- Next Key Share
- Update
- Next Key Weak Exclusive
- Weak Exclusive

Vergleichselemente

Es gibt zwei Angaben, die Informationen über die in einem Zugriffsplan verwendeten Vergleichselemente enthalten:

1. Die folgende Angabe zeigt die Anzahl von Vergleichselementen, die für jeden Datenblock ausgewertet werden, der aus einem Blockindex abgerufen wird.

   ```
   Block Predicates(s)
   |  #Predicates = n
   ```

2. Die folgende Angabe zeigt die Anzahl von Vergleichselementen, die ausgewertet werden, während auf die Daten zugegriffen wird. Die Anzahl der Vergleichselemente lässt verschobene (Pushdown) Operationen wie Spaltenberechnung oder Sortierung unberücksichtigt.

   ```
   Sargable Predicate(s)
   |  #Predicates = n
   ```

3. Die folgende Angabe zeigt die Anzahl von Vergleichselementen, die ausgewertet werden, nachdem die Daten zurückgegeben wurden (d. h. Restvergleichselemente):

```
Residual Predicate(s)
|  #Predicates = n
```

Die Anzahl der in den obigen Angaben gezeigten Vergleichselemente spiegelt möglicherweise die Anzahl der Vergleichselemente der SQL-Anweisung aus folgenden Gründen nicht genau wider:

- Vergleichselemente können mehrmals in derselben Abfrage verwendet werden.
- Vergleichselemente können durch Hinzufügung impliziter Vergleichselemente während der Optimierung der Abfrage umgewandelt und erweitert worden sein.
- Vergleichselemente können während der Optimierung der Abfrage in weniger Vergleichselemente umgewandelt und komprimiert worden sein.

Verschiedene Tabellenangaben

- Die folgende Angabe weist darauf hin, dass nur auf eine Zeile zugegriffen wird:

  ```
  Single Record
  ```

- Die folgende Angabe wird angezeigt, wenn die für diesen Tabellenzugriff verwendete Isolationsstufe nicht der Isolationsstufe der Anweisung entspricht:

  ```
  Isolation Level: xxxx
  ```

  Eine andere Isolationsstufe kann aus einer Reihe von Gründen verwendet werden, zum Beispiel:

  – Ein Paket wurde mit wiederholtem Lesen (RR) gebunden und hat Auswirkungen auf referenzielle Integritätsbedingungen. Der Zugriff der übergeordneten Tabelle zum Prüfen referenzieller Integritätsbedingungen wird auf die Isolationsstufe Cursorstabilität (CS) heruntergestuft, um unnötige Sperren für diese Tabelle zu vermeiden.

  – Ein mit der Isolationsstufe für nicht festgeschriebenen Lesevorgang (UR) gebundenes Paket gibt eine Anweisung DELETE oder UPDATE aus. Der Tabellenzugriff für den tatsächlichen Löschvorgang wird auf Cursorstabilität (CS) hochgestuft.

- Die folgende Angabe weist darauf hin, dass einige oder alle Zeilen, die aus der temporären Tabelle gelesen wurden, außerhalb des Pufferpools zwischengespeichert werden, wenn genügend Sortierspeicher verfügbar ist:

  ```
  Keep Rows In Private Memory
  ```

- Wenn für die Tabelle das Attribut für flüchtige Kardinalität definiert wurde, wird dies folgendermaßen angezeigt:

  ```
  Volatile Cardinality
  ```

Zugehörige Konzepte:

- „Beschreibung der Ausgabe von db2expln und dynexpln" auf Seite 645
- „Beispiele für die Ausgabe von db2expln und dynexpln" auf Seite 665

Informationen zu temporären Tabellen

Eine temporäre Tabelle wird von einem Zugriffsplan während dessen Ausführung zum Speichern von Daten in einer temporären Arbeitstabelle oder Übergangstabelle verwendet. Die Tabelle existiert nur, solange der Zugriffsplan ausgeführt wird. Allgemein werden temporäre Tabellen verwendet, wenn Unterabfragen frühzeitig im Zugriffsplan ausgewertet werden müssen oder wenn Zwischenergebnisse nicht in den vorhandenen Speicher passen.

Wenn eine temporäre Tabelle erstellt werden muss, kann eine von zwei möglichen Angaben angezeigt werden. Diese Angaben weisen darauf hin, dass eine temporäre Tabelle erstellt und Zeilen in sie eingefügt werden. Die ID ist eine Kennung, die aus praktischen Gründen von `db2expln` zugeordnet wird, wenn auf die temporäre Tabelle Bezug genommen wird. Dieser ID wird als Präfix der Buchstabe 't' vorangestellt, um anzuzeigen, dass es sich um eine temporäre Tabelle handelt.

- Die folgende Angabe bedeutet, dass eine normale temporäre Tabelle erstellt wird:
  ```
  Insert Into Temp Table  ID = tn
  ```
- Die folgende Angabe bedeutet, dass eine normale temporäre Tabelle von mehreren Subagenten parallel erstellt wird:
  ```
  Insert Into Shared Temp Table  ID = tn
  ```
- Die folgende Angabe bedeutet, dass eine sortierte temporäre Tabelle erstellt wird:
  ```
  Insert Into Sorted Temp Table  ID = tn
  ```
- Die folgende Angabe bedeutet, dass eine sortierte temporäre Tabelle von mehreren Subagenten parallel erstellt wird:
  ```
  Insert Into Sorted Shared Temp Table  ID = tn
  ```
- Die folgende Angabe bedeutet, dass eine deklarierte globale temporäre Tabelle erstellt wird:
  ```
  Insert Into Global Temp Table  ID = ts,tn
  ```
- Die folgende Angabe bedeutet, dass eine deklarierte globale temporäre Tabelle von mehreren Subagenten parallel erstellt wird:
  ```
  Insert Into Shared Global Temp Table  ID = ts,tn
  ```
- Die folgende Angabe bedeutet, dass eine sortierte deklarierte globale temporäre Tabelle erstellt wird:
  ```
  Insert Into Sorted Global Temp Table  ID = ts,tn
  ```
- Die folgende Angabe bedeutet, dass eine sortierte deklarierte globale temporäre Tabelle von mehreren Subagenten parallel erstellt wird:
  ```
  Insert Into Sorted Shared Global Temp Table  ID = ts,tn
  ```

Nach jeder der oben gezeigten Angaben folgt die Angabe:

```
#Columns = n
```

Diese Angabe zeigt die Anzahl der Spalten für jede Zeile, die in die temporäre Tabelle eingefügt wird.

Sortierte temporäre Tabellen

Sortierte temporäre Tabellen treten z. B. als Ergebnis folgender Operationen auf:

- ORDER BY
- DISTINCT
- GROUP BY

- Mischverknüpfung (Merge Join)
- '= ANY' Unterabfrage
- '<> ALL' Unterabfrage
- INTERSECT oder EXCEPT
- UNION (ohne das Schlüsselwort ALL)

Eine Reihe zusätzlicher Angaben kann auf die ursprüngliche Angabe über die Erstellung für eine sortierte temporäre Tabelle folgen:

- Die folgende Angabe zeigt die Anzahl der bei der Sortierung verwendeten Schlüsselspalten:

  ```
  #Sort Key Columns = n
  ```

  Für jede Spalte im Sortierschlüssel wird eine der folgenden Zeilen angezeigt:

  ```
  Key n: spaltenname (Ascending)
  Key n: spaltenname (Descending)
  Key n: (Ascending)
  Key n: (Descending)
  ```

- Die folgenden Angaben enthalten Schätzwerte für die Anzahl und die Größe der Zeilen, so dass die optimale Größe für den Sortierspeicher zur Laufzeit zugeordnet werden kann.

  ```
  Sortheap Allocation Parameters:
  │ #Rows     = n
  │ Row Width = n
  ```

- Wenn lediglich die ersten Zeilen des sortierten Ergebnisses benötigt werden, wird Folgendes angezeigt:

  ```
  Sort Limited To Estimated Row Count
  ```

- Für Sortiervorgänge in einer symmetrischen Mehrprozessorumgebung (SMP-Umgebung) wird die Art der durchzuführenden Sortierung durch eine der folgenden Angaben angezeigt:

  ```
  Use Partitioned Sort
  Use Shared Sort
  Use Replicated Sort
  Use Round-Robin Sort
  ```

- Die folgenden Angaben zeigen an, ob das Ergebnis der Sortierung im Sortierspeicher verbleibt:

  ```
      Piped
  ```

  und

  ```
  Not Piped
  ```

  Wenn eine über Pipe geleitete Sortierung angegeben wird, behält der Datenbankmanager die sortierte Ausgabe im Speicher, anstatt das sortierte Ergebnis in eine andere temporäre Tabelle zu schreiben.

- Die folgende Angabe bedeutet, dass doppelte Werte während der Sortierung entfernt werden:

  ```
  Duplicate Elimination
  ```

- Wenn Spaltenberechnungen in der Sortierung durchgeführt werden, wird dies durch eine der folgenden Angaben angezeigt:

  ```
  Partial Aggregation
  Intermediate Aggregation
  Buffered Partial Aggregation
  Buffered Intermediate Aggregation
  ```

Abschließen der temporären Tabelle: Nach einem Tabellenzugriff, der eine verschobene (Pushdown) Operation zum Erstellen einer temporären Tabelle enthält (d. h. eine Operation zum Erstellen einer temporären Tabelle, die im Rahmen eines Tabellenzugriffs auftritt), folgt eine Abschlussangabe (Completion), die das Dateiende verarbeitet, indem sie die temporäre Tabelle darauf vorbereitet, Zeilen für den nachfolgenden Zugriff auf die temporäre Tabelle zur Verfügung zu stellen. Es wird eine der folgenden Zeilen angezeigt:

```
Temp Table Completion  ID = tn
Shared Temp Table Completion  ID = tn
Sorted Temp Table Completion  ID = tn
Sorted Shared Temp Table Completion  ID = tn
```

Tabellenfunktionen

Tabellenfunktionen sind benutzerdefinierte Funktionen (UDFs), die Daten in Form einer Tabelle an die Anweisung zurückgeben. Tabellenfunktionen werden durch folgende Angaben angezeigt:

```
Access User Defined Table Function
|  Name = schema.funkname
|  Specific Name = spezifischername
|  SQL Access Level = zugriffsebene
|  Language = sprache
|  Parameter Style = parmstil
|  Fenced                      Not Deterministic
|  Called on NULL Input        Disallow Parallel
|  Not Federated               Not Threadsafe
```

Der spezifische Name identifiziert die aufgerufene Tabellenfunktion eindeutig. Die übrigen Zeilen geben Details über die Attribute der Funktion an.

Zugehörige Konzepte:

- „Beschreibung der Ausgabe von db2expln und dynexpln" auf Seite 645
- „Beispiele für die Ausgabe von db2expln und dynexpln" auf Seite 665

Verknüpfungsinformationen

Es gibt drei Arten von Verknüpfungen:

- Hashverknüpfung (Hash Join)
- Mischverknüpfung (Merge Join)
- Verknüpfung über Verschachtelungsschleife (Nested Loop Join)

Wenn bei der Ausführung eines Abschnitts der Zeitpunkt kommt, zu dem eine Verknüpfung ausgeführt wird, wird eine der folgenden Angaben angezeigt:

```
Hash Join
Merge Join
Nested Loop Join
```

Es ist möglich, dass eine linke äußere Verknüpfung durchgeführt wird. Eine linke äußere Verknüpfung wird durch eine der folgenden Angaben angezeigt:

```
Left Outer Hash Join
Left Outer Merge Join
Left Outer Nested Loop Join
```

Bei Mischverknüpfungen und Verknüpfungen über Verschachtelungsschleife ist die äußere Tabelle der Verknüpfung die Tabelle, auf die in der vorherigen Zugriffsangabe, die in der Ausgabe angezeigt wird, verwiesen wird. Die innere Tabelle bei dieser Verknüpfung ist die Tabelle, auf die in der Zugriffsangabe verwiesen wird,

die sich im Bereich der Verknüpfungsangabe befindet. Bei Hashverknüpfungen sind die Zugriffsangaben umgekehrt, d. h. die äußere Tabelle ist im Verknüpfungsbereich enthalten, und die innere Tabelle wird vor der Verknüpfung angezeigt.

Bei einer Hash- oder Mischverknüpfung können folgende weitere Angaben auftreten:

- In einigen Fällen muss bei einer Verknüpfung lediglich festgestellt werden, ob eine Zeile der inneren Tabelle mit der aktuellen Zeile in der äußeren Tabelle übereinstimmt. Dies wird durch folgende Angabe angezeigt:

  ```
  Early Out: Single Match Per Outer Row
  ```

- Es ist möglich, nach Abschluss der Verknüpfung Vergleichselemente anzuwenden. Die Anzahl der Vergleichselemente, die angewandt werden, wird folgendermaßen angezeigt:

  ```
  Residual Predicate(s)
  |  #Predicates = n
  ```

Bei einer Hashverknüpfung können folgende weitere Angaben auftreten:

- Die Hashtabelle wird aus der inneren Tabelle erstellt. Wenn die Erstellung der Hashtabelle in ein Vergleichselement im Zugriff auf die innere Tabelle verschoben wurde, wird darauf durch die folgende Angabe im Zugriff der inneren Tabelle hingewiesen:

  ```
  Process Hash Table For Join
  ```

- Beim Zugriff der äußeren Tabelle kann eine Prüftabelle erstellt werden, um die Leistung der Verknüpfung zu steigern. Auf die Erstellung der Prüftabelle wird durch die folgende Angabe im Zugriff der äußeren Tabelle hingewiesen:

  ```
  Process Probe Table For Hash Join
  ```

- Die geschätzte erforderliche Anzahl Byte zum Erstellen der Hashtabelle wird durch folgende Angabe dargestellt:

  ```
  Estimated Build Size: n
  ```

- Die geschätzte erforderliche Anzahl Byte für die Prüftabelle wird durch folgende Angabe dargestellt:

  ```
  Estimated Probe Size: n
  ```

Bei einer Verknüpfung über Verschachtelungsschleife kann die folgende zusätzliche Angabe direkt nach der Verknüpfungsangabe angezeigt werden:

```
Piped Inner
```

Diese Angabe bedeutet, dass die innere Tabelle der Verknüpfung das Ergebnis einer anderen Reihe von Operationen ist. Dies wird auch als eine *zusammengesetzte innere (composite inner)* Tabelle bezeichnet.

Wenn eine Verknüpfung mehr als zwei Tabellen umfasst, müssen die EXPLAIN-Schritte von oben nach unten gelesen werden. Angenommen, die EXPLAIN-Ausgabe hat folgende Struktur:

```
Access ..... W
Join
|  Access ..... X
Join
|  Access ..... Y
Join
|  Access ..... Z
```

Die Ausführung würde in diesem Fall in folgenden Schritten ablaufen:

1. Abrufen einer den Bedingungen entsprechenden Zeile aus `W`

2. Verknüpfen der Zeile aus `W` mit der (nächsten) Zeile aus `X` und Benennung des Ergebnisses als `P1` (für Verknüpfungsteilergebnis Nr. 1)
3. Verknüpfung von `P1` mit der (nächsten) Zeile aus `Y` zum Erstellen von `P2`
4. Verknüpfung von `P2` mit der (nächsten) Zeile aus `Z` zum Erstellen einer vollständigen Ergebniszeile
5. Wenn weitere Zeilen in `Z` sind, weiter mit Schritt 4
6. Wenn weitere Zeilen in `Y` sind, weiter mit Schritt 3
7. Wenn weitere Zeilen in `X` sind, weiter mit Schritt 2
8. Wenn weitere Zeilen in `W` sind, weiter mit Schritt 1

Zugehörige Konzepte:

- „Beschreibung der Ausgabe von db2expln und dynexpln" auf Seite 645
- „Beispiele für die Ausgabe von db2expln und dynexpln" auf Seite 665

Datenstrominformationen

In einem Zugriffsplan ist es oft erforderlich, die Erstellung und den Fluss von Daten von einer Reihe von Operationen zur anderen zu steuern. Das Konzept des Datenstroms erlaubt es, eine Gruppe von Operationen innerhalb eines Zugriffsplans als Einheit zu steuern. Der Beginn eines Datenstroms wird durch folgende Angabe gekennzeichnet:

```
Data Stream n
```

Hierbei ist `n` eine eindeutige Kennung, die zur leichteren Bezugnahme von `db2expln` zugeordnet wird. Das Ende des Datenstroms wird durch folgende Angabe gekennzeichnet:

```
End of Data Stream n
```

Alle Operationen zwischen diesen Angaben werden als Teil desselben Datenstroms angesehen.

Ein Datenstrom hat eine Anzahl von Merkmalen, und auf die einleitende Datenstromangabe können eine oder mehrere Angaben folgen, um diese Merkmale zu beschreiben:

- Wenn die Verarbeitung des Datenstroms von einem Wert abhängt, der früher im Zugriffsplan generiert wurde, wird der Datenstrom mit folgender Angabe markiert:

  ```
  Correlated
  ```

- Ähnlich wie bei einer sortierten temporären Tabelle zeigen die folgenden Angaben, ob die Ergebnisse des Datenstroms im Speicher behalten werden:

  ```
  Piped
  ```

  und

  ```
  Not Piped
  ```

  Wie bei temporären Tabellen kann ein über eine Pipe geleiteter Datenstrom auf die Platte geschrieben werden, wenn zur Ausführungszeit zu wenig Speicher vorhanden ist. Der Zugriffsplan sieht beide Möglichkeiten vor.

- Die folgende Angabe bedeutet, dass nur ein einziger Satz aus diesem Datenstrom benötigt wird:

  ```
  Single Record
  ```

Wenn auf einen Datenstrom zugegriffen wird, wird die folgende Angabe in der Ausgabe angezeigt:

```
Access Data Stream n
```

Zugehörige Konzepte:

- „Beschreibung der Ausgabe von db2expln und dynexpln" auf Seite 645
- „Beispiele für die Ausgabe von db2expln und dynexpln" auf Seite 665

Informationen zu INSERT, UPDATE und DELETE

Der EXPLAIN-Text für diese SQL-Anweisungen ist selbsterklärend. Nachfolgend sind mögliche Texte für diese SQL-Operationen dargestellt:

```
Insert: Table Name = schema.name ID = ts,n
Update: Table Name = schema.name ID = ts,n
Delete: Table Name = schema.name ID = ts,n
Insert: Hierarchy Table Name = schema.name ID = ts,n
Update: Hierarchy Table Name = schema.name ID = ts,n
Delete: Hierarchy Table Name = schema.name ID = ts,n
Insert: Materialized Query Table = schema.name ID = ts,n
Update: Materialized Query Table = schema.name ID = ts,n
Delete: Materialized Query Table = schema.name ID = ts,n
Insert: Global Temporary Table ID = ts, tn
Update: Global Temporary Table ID = ts, tn
Delete: Global Temporary Table ID = ts, tn
```

Zugehörige Konzepte:

- „Beschreibung der Ausgabe von db2expln und dynexpln" auf Seite 645
- „Beispiele für die Ausgabe von db2expln und dynexpln" auf Seite 665

Informationen zur Vorbereitung von Block- und Zeilen-IDs

Für einige Zugriffspläne ist es effizienter, wenn die den Bedingungen entsprechenden Satz-IDs (RIDs) und Block-IDs (BIDs) sortiert und mehrfach auftretende Werte entfernt werden (bei OR-Verknüpfung von Indizes) bzw. wenn eine Technik verwendet wird, mit der IDs, die in allen verwendeten Indizes auftreten (bei AND-Verknüpfung von Indizes), identifiziert werden, bevor der tatsächliche Zugriff auf die Tabelle erfolgt. Es gibt drei Hauptgründe für die Vorbereitung von IDs, auf die in den EXPLAIN-Angaben hingewiesen wird:

- Die beiden folgenden Angaben bedeuten, dass eine OR-Verknüpfung von Indizes (Index ORing) zur Vorbereitung der Liste der qualifizierten IDs verwendet wird:

  ```
  Index ORing Preparation
  Block Index ORing Preparation
  ```

  Index ORing (OR-Verknüpfung von Indizes) bezeichnet ein Verfahren, bei dem mehrere Indexzugriffe vorgenommen und die Ergebnisse kombiniert werden, so dass die unterschiedlichen IDs, die mindestens in einem der Indizes auftreten, auf die zugegriffen wird, zusammengefasst werden. Das Optimierungsprogramm erwägt die Verwendung dieses Verfahrens, wenn Vergleichselemente durch Schlüsselwörter OR verknüpft werden oder ein Vergleichselement IN existiert. Die Zugriffe können auf den gleichen Index oder auf verschiedene Indizes erfolgen.

- Ein weiterer Grund für die Vorbereitung von IDs ist die Vorbereitung der Eingabedaten, die während des Vorablesezugriffs über Listen verwendet werden sollen, wie dies durch folgende Angaben angezeigt wird:

  ```
  List Prefetch Preparation
  Block List Prefetch RID Preparation
  ```

- *Index ANDing* (AND-Verknüpfung von Indizes) bezeichnet ein Verfahren, beim dem mehrere Indexzugriffe vorgenommen und die Ergebnisse kombiniert werden, so dass nur die IDs zusammengefasst werden, die in allen Indizes auftreten, auf die zugegriffen wird. Die folgenden Angaben zeigen an, dass die Verarbeitung einer AND-Verknüpfung von Indizes gestartet wird:

  ```
  Index ANDing
  Block Index ANDing
  ```

  Wenn das Optimierungsprogramm die Größe der Ergebnismenge abgeschätzt hat, wird der Schätzwert mit der folgenden Angabe angezeigt:

  ```
  Optimizer Estimate of Set Size: n
  ```

  Die Filteroperationen bei AND-Verknüpfungen von Indizes verarbeiten die IDs und verwenden Bit-Filtermethoden, um die IDs zu bestimmen, die in allen Indizes vorkommen, auf die zugegriffen wird. Die folgenden Angaben weisen darauf hin, dass IDs zur AND-Verknüpfung von Indizes verarbeitet werden:

  ```
  Index ANDing Bitmap Build Using Row IDs
  Index ANDing Bitmap Probe Using Row IDs
  Index ANDing Bitmap Build and Probe Using Row IDs
  Block Index ANDing Bitmap Build Using Block IDs
  Block Index ANDing Bitmap Build and Probe Using Block IDs
  Block Index ANDing Bitmap Build and Probe Using Row IDs
  Block Index ANDing Bitmap Probe Using Block IDs and Build Using Row IDs
  Block Index ANDing Bitmap Probe Using Block IDs
  Block Index ANDing Bitmap Probe Using Row IDs
  ```

  Wenn das Optimierungsprogramm die Größe der Ergebnismenge für eine Bitmaske (Bitmap) abgeschätzt hat, wird der Schätzwert mit der folgenden Angabe angezeigt:

  ```
  Optimizer Estimate of Set Size: n
  ```

Bei jeder Art von ID-Vorbereitung wird die Möglichkeit, dass ein Vorablesezugriff über Listen erfolgen kann, mit folgender Angabe angezeigt:

```
Prefetch: Enabled
```

Zugehörige Konzepte:

- „Beschreibung der Ausgabe von db2expln und dynexpln" auf Seite 645
- „Beispiele für die Ausgabe von db2expln und dynexpln" auf Seite 665

Informationen zu Spaltenberechnungen (Aggregation)

Spaltenberechnungen (Aggregation) werden für die Zeilen vorgenommen, die den etwaigen, in den Vergleichselementen der SQL-Anweisungen festgelegten Bedingungen entsprechen. Wenn eine Art von Spaltenfunktion auszuführen ist, wird eine der folgenden Angaben angezeigt:

```
Aggregation
Predicate Aggregation
Partial Aggregation
Partial Predicate Aggregation
```

```
Intermediate Aggregation
Intermediate Predicate Aggregation
Final Aggregation
Final Predicate Aggregation
```

Die Angabe *Predicate aggregation* besagt, dass die Spaltenberechnungsoperation zur Verarbeitung als Vergleichselement auf den Zeitpunkt, zu dem auf die Daten wirklich zugegriffen wird, verschoben wurde.

Unter jeder der oben aufgeführten Angaben über Spaltenberechnungen befindet sich eine Angabe, die die Art der durchgeführten Spaltenfunktion anzeigt:

```
Group By
Column Function(s)
Single Record
```

Die spezifische Spaltenfunktion kann von der ursprünglichen SQL-Anweisung abgeleitet werden. Ein einzelner Satz (Single Record) wird aus einem Index abgerufen, um eine Operation MIN oder MAX auszuführen.

Wenn eine Spaltenberechnung über Vergleichselemente verwendet wird, folgt auf die Angabe über den Tabellenzugriff, in der die Spaltenberechnung auftrat, eine Angabe über den Abschluss der Spaltenberechnung, die alle erforderlichen Verarbeitungsschritte bei Gruppenende bzw. bei Dateiende vornimmt. Dazu wird eine der folgenden Zeilen in der Ausgabe angezeigt:

```
Aggregation Completion
Partial Aggregation Completion
Intermediate Aggregation Completion
Final Aggregation Completion
```

Zugehörige Konzepte:

- „Beschreibung der Ausgabe von db2expln und dynexpln" auf Seite 645
- „Beispiele für die Ausgabe von db2expln und dynexpln" auf Seite 665

Informationen zur Parallelverarbeitung

Für die parallele Ausführung einer SQL-Anweisung (entweder mit partitionsinterner oder partitionsübergreifender Parallelität) sind einige besondere Operationen erforderlich. Die Operationen für Parallelpläne werden im Folgenden beschrieben.

- Bei der Ausführung eines partitionsinternen Parallelplans werden Abschnitte des Plans gleichzeitig mit Hilfe verschiedener Subagenten ausgeführt. Die Erstellung der Subagenten wird durch folgende Angabe angezeigt:

  ```
  Process Using n Subagents
  ```

- Bei der Ausführung eines partitionsübergreifenden parallelen Zugriffsplans wird der Abschnitt (Section) in mehrere Teilbereiche (Subsections) geteilt. Jeder Teilbereich wird zur Ausführung an einen oder mehrere Knoten gesendet. Ein wichtiger Teilbereich ist der *Koordinatorteilbereich*. Der Koordinatorteilbereich ist der erste Teilbereich in jedem Plan. Er erhält zunächst die Steuerung und ist dann für die Verteilung der anderen Teilbereiche und die Rückgabe von Ergebnissen an die aufrufende Anwendung zuständig.

  Die Verteilung von Teilbereichen wird durch folgende Angabe angezeigt:

  ```
  Distribute Subsection #n
  ```

  Die Knoten, die einen Teilbereich empfangen, können durch eine von acht Methoden bestimmt werden:

- Die folgende Angabe bedeutet, dass der Teilbereich abhängig vom Wert der Spalten zu einem Knoten innerhalb der Datenbankpartitionsgruppe gesendet wird.

  ```
  Directed by Hash
  |  #Columns = n
  |  Partition Map ID = n, Nodegroup = ngname, #Nodes = n
  ```

- Die folgende Angabe bedeutet, dass der Teilbereich an einen vorbestimmten Knoten gesendet wird. (Dies tritt häufig auf, wenn die Anweisung mit der Funktion NODENUMBER() arbeitet.)

  ```
  Directed by Node Number
  ```

- Die folgende Angabe bedeutet, dass der Teilbereich an den Knoten gesendet wird, der einer vorbestimmten Partitionsnummer in der angegebenen Datenbankpartitionsgruppe entspricht. (Dies tritt häufig auf, wenn die Anweisung mit der Funktion PARTITION() arbeitet.)

  ```
  Directed by Partition Number
  |  Partition Map ID = n, Nodegroup = ngname, #Nodes = n
  ```

- Die folgende Angabe bedeutet, dass der Teilbereich an den Knoten gesendet wird, der die aktuelle Zeile für den Cursor der Anwendung verfügbar gemacht hat.

  ```
  Directed by Position
  ```

- Die folgende Angabe bedeutet, dass nur ein einziger Knoten, der bei der Kompilierung der Anweisung festgelegt wird, den Teilbereich empfängt.

  ```
  Directed to Single Node
  |  Node Number = n
  ```

- Die folgenden Angaben bedeuten, dass der Teilbereich auf dem Koordinatorknoten ausgeführt wird.

  ```
  Directed to Application Coordinator Node
  Directed to Local Coordinator Node
  ```

- Die folgende Angabe bedeutet, dass der Teilbereich an alle aufgeführten Knoten gesendet wird.

  ```
  Broadcast to Node List
  |  Nodes = n1, n2, n3, ...
  ```

- Die folgende Angabe bedeutet, dass nur ein einziger Knoten, der bei der Ausführung der Anweisung festgelegt wird, den Teilbereich empfängt.

  ```
  Directed to Any Node
  ```

- Tabellenwarteschlangen werden zum Versetzen von Daten zwischen Teilbereichen in einer Umgebung mit partitionierten Datenbanken oder zwischen Subagenten in einer symmetrischen Mehrprozessorumgebung (SMP) verwendet. Tabellenwarteschlangen werden folgendermaßen beschrieben:

  - Die folgenden Angaben zeigen an, dass Daten in eine Tabellenwarteschlange eingefügt werden:

    ```
    Insert Into Synchronous Table Queue  ID = qn
    Insert Into Asynchronous Table Queue  ID = qn
    Insert Into Synchronous Local Table Queue  ID = qn
    Insert Into Asynchronous Local Table Queue  ID = qn
    ```

  - Bei Tabellenwarteschlangen einer Datenbankpartition wird das Ziel für Zeilen, die in die Tabellenwarteschlange eingefügt werden, durch eine der folgenden Angaben angezeigt:

    Alle Zeilen werden an den Koordinatorknoten gesendet:

    ```
    Broadcast to Coordinator Node
    ```

    Alle Zeilen werden an jede Datenbankpartition gesendet, auf der der angegebene Teilbereich ausgeführt wird:

```
Broadcast to All Nodes of Subsection n
```

Jede Zeile wird abhängig von den Werten in der Zeile an eine Datenbankpartition gesendet:

```
Hash to Specific Node
```

Jede Zeile wird an eine Datenbankpartition gesendet, die während der Ausführung der Anweisung bestimmt wird:

```
Send to Specific Node
```

Jede Zeile wird an einen zufällig bestimmten Knoten gesendet:

```
Send to Random Node
```

– In einigen Fällen ist es möglich, dass die Tabellenwarteschlange der Datenbankpartition einige Zeilen wegen Kapazitätsüberschreitung in eine temporäre Tabelle versetzen muss. Diese Möglichkeit wird durch folgende Angabe angezeigt:

```
Rows Can Overflow to Temporary Table
```

– Auf einen Tabellenzugriff, der eine verschobene Operation (Pushdown) zum Einfügen von Zeilen in eine Tabellenwarteschlange enthält, folgt eine Abschlussangabe (Completion), die die Zeilen handhabt, die nicht unmittelbar gesendet werden konnten. Dazu wird eine der folgenden Zeilen in der Ausgabe angezeigt:

```
Insert Into Synchronous Table Queue Completion  ID = qn
Insert Into Asynchronous Table Queue Completion  ID = qn
Insert Into Synchronous Local Table Queue Completion  ID = qn
Insert Into Asynchronous Local Table Queue Completion  ID = qn
```

– Die folgenden Angaben zeigen an, dass Daten aus einer Tabellenwarteschlange abgerufen werden:

```
Access Table Queue  ID = qn
Access Local Table Queue  ID = qn
```

Diesen Nachrichten folgt stets eine Angabe der Anzahl der Zeilen, die abgerufen werden.

```
#Columns = n
```

– Wenn die Tabellenwarteschlange die Zeilen auf der Empfängerseite sortiert, wird der Angabe über den Tabellenwarteschlangenzugriff außerdem eine der folgenden Nachrichten beigefügt:

```
Output Sorted
Output Sorted and Unique
```

Diesen Nachrichten folgt eine Angabe der Anzahl der Schlüsselspalten, die für die Sortierung verwendet wurden.

```
#Key Columns = n
```

Für jede Spalte im Sortierschlüssel wird eine der folgenden Angaben angezeigt:

```
Key n: (Ascending)
Key n: (Descending)
```

– Wenn Vergleichselemente auf der Empfängerseite der Tabellenwarteschlange auf die Zeilen angewendet werden, wird folgende Nachricht in der Ausgabe angezeigt:

```
Residual Predicate(s)
|  #Predicates = n
```

- Einige Teilbereiche in Umgebungen mit partitionierten Datenbanken springen explizit zurück an den Start des Teilbereichs. Dies wird durch folgende Angabe angezeigt:

  ```
  Jump Back to Start of Subsection
  ```

Zugehörige Konzepte:

- „Beschreibung der Ausgabe von db2expln und dynexpln" auf Seite 645
- „Beispiele für die Ausgabe von db2expln und dynexpln" auf Seite 665

Informationen zu Abfragen auf zusammengeschlossene Datenbanken

Für die Ausführung einer SQL-Anweisung in einer zusammengeschlossenen Datenbank ist die Fähigkeit erforderlich, Teile der betreffenden Anweisung an anderen Datenquellen auszuführen.

Die folgende Angabe zeigt, dass eine Datenquelle gelesen wird:

```
Ship Distributed Subquery #n
|  #Columns = n
```

Es ist möglich, dass Vergleichselemente auf die Daten angewandt werden, die von der verteilten Unterabfrage (Distributed Subquery) zurückgegeben werden. Die Anzahl der Vergleichselemente (Predicates), die angewandt werden, wird folgendermaßen angezeigt:

```
Residual Predicate(s)
|   #Predicates = n
```

Eine INSERT-, UPDATE- oder DELETE-Operation, die an einer Datenquelle stattfindet, wird durch die entsprechende Angabe gezeigt:

```
Ship Distributed Insert #n
Ship Distributed Update #n
Ship Distributed Delete #n
```

Wenn eine Tabelle an einer Datenquelle explizit gesperrt wird, wird dies durch folgende Angabe gezeigt:

```
Ship Distributed Lock Table #n
```

DDL-Anweisungen für eine Datenquelle werden in zwei Teile aufgeteilt. Der Teil, der an der Datenquelle aufgerufen wird, wird wie folgt gezeigt:

```
Ship Distributed DDL Statement #n
```

Wenn der Server der zusammengeschlossenen Datenbanken eine partitionierte Datenbank ist, muss ein Teil der DDL-Anweisung auf dem Katalogknoten ausgeführt werden. Dies wird wie folgt angegeben:

```
Distributed DDL Statement #n Completion
```

Die Einzelheiten für die verteilten Unteranweisungen werden getrennt bereitgestellt. Die Optionen für verteilte Unteranweisungen werden nachfolgend beschrieben:

- Die Datenquelle für die Unterabfrage wird durch eine der folgenden Angaben gezeigt:

  ```
  Server: servername (typ, version)
  Server: servername (typ)
  Server: servername
  ```

- Wenn die Datenquelle relational ist, wird das SQL für die Unteranweisung wie folgt dargestellt:

  ```
  SQL Statement:
  anweisung
  ```

  Nicht relationale Datenquellen werden wie folgt angegeben:

  ```
  Non-Relational Data Source
  ```

- Die Kurznamen (Nicknames), auf die in der Unteranweisung verwiesen wird, werden folgendermaßen aufgelistet:

  ```
  Nicknames Referenced:
     schema.kurzname ID = n
  ```

  Wenn die Datenquelle relational ist, wird die Basistabelle für den Kurznamen wie folgt angegeben:

  ```
  Base = basisschema.basistabelle
  ```

  Wenn die Datenquelle nicht relational ist, wird die Quellendatei für den Kurznamen wie folgt angezeigt:

  ```
  Source File = dateiname
  ```

- Wenn die Werte vom Server mit zusammengeschlossenen Datenbanken an die Datenquelle übergeben werden, bevor die Unteranweisung ausgeführt wird, wird die Zahl der Werte folgendermaßen gezeigt:

  ```
  #Input Columns: n
  ```

- Wenn die Werte von der Datenquelle an den Server mit zusammengeschlossenen Datenbanken übergeben werden, nachdem die Unteranweisung ausgeführt wird, wird die Zahl der Werte folgendermaßen gezeigt:

  ```
  #Output Columns: n
  ```

Zugehörige Konzepte:

- „Richtlinien zur Analyse, wo eine Abfrage für zusammengeschlossene Datenbanken ausgewertet wird" auf Seite 215
- „Beschreibung der Ausgabe von db2expln und dynexpln" auf Seite 645

Verschiedene Informationen

- Abschnitte für Anweisungen der Datendefinitionssprache (DDL - Data Definition Language) werden in der Ausgabe folgendermaßen gekennzeichnet:

  ```
  DDL Statement
  ```

  Für DDL-Anweisungen wird keine weitere EXPLAIN-Ausgabe bereitgestellt.

- Abschnitte für SET-Anweisungen für die aktualisierbaren Sonderregister wie **CURRENT EXPLAIN SNAPSHOT** werden in der Ausgabe folgendermaßen gekennzeichnet:

  ```
  SET Statement
  ```

  Für SET-Anweisungen wird keine weitere EXPLAIN-Ausgabe bereitgestellt.

- Wenn die SQL-Anweisung die Klausel DISTINCT enthält, kann der folgende Text in der Ausgabe angezeigt werden:

  ```
  Distinct Filter  #Columns = n
  ```

  Dabei ist `n` die Anzahl von Spalten, die beim Abrufen eindeutiger Zeilen verwendet wird. Zum Abrufen eindeutiger Zeilenwerte müssen die Zeilen geordnet sein, so dass gleiche Werte übersprungen werden können. Diese Angabe wird nicht angezeigt, wenn der Datenbankmanager gleiche Werte nicht explizit entfernen muss wie in folgenden Fällen:

  - Es existiert ein eindeutiger Index, und alle Spalten im Indexschlüssel sind Teil der Operation für die Klausel DISTINCT.
  - Die mehrfach auftretenden Werte können beim Sortieren entfernt werden.
- Die folgende Angabe wird angezeigt, wenn die nächste Operation von einer bestimmten Satz-ID (RID) abhängig ist:

  ```
  Positioned Operation
  ```

  Wenn die Positionierungsoperation an einer Datenquelle in einer zusammengeschlossenen Datenbankumgebung stattfindet, sieht die Angabe wie folgt aus:

  ```
  Distributed Positioned Operation
  ```

  Diese Angabe wird für jede SQL-Anweisung angezeigt, die die Syntax WHERE CURRENT OF verwendet.
- Die folgende Angabe wird angezeigt, wenn Vergleichselemente vorhanden sind, die auf das Ergebnis angewendet werden müssen, aber die nicht als Teil einer anderen Operation angewendet werden konnten (Restvergleichselemente):

  ```
  Residual Predicate Application
  |  #Predicates = n
  ```
- Die folgende Angabe wird angezeigt, wenn ein Operator UNION in der SQL-Anweisung enthalten ist:

  ```
  UNION
  ```
- Die folgende Angabe wird angezeigt, wenn sich im Zugriffsplan eine Operation befindet, deren einziger Zweck das Erstellen von Zeilenwerten zur Verwendung durch nachfolgende Operationen ist:

  ```
  Table Constructor
  |  n-Row(s)
  ```

  Table Constructors können verwendet werden, um Werte in einer Menge in eine Reihe von Zeilen umzuwandeln, die anschließend an nachfolgende Operationen übergeben werden. Wenn die nächste Zeile von einem Table Constructor angefordert wird, wird die folgende Angabe angezeigt:

  ```
  Access Table Constructor
  ```
- Die folgende Angabe wird angezeigt, wenn eine Operation existiert, die nur unter bestimmten Bedingungen verarbeitet wird:

  ```
  Conditional Evaluation
  |  Condition #n:
  |  #Predicates = n
  |  Action #n:
  ```

  Durch bedingte Auswertung werden Aktivitäten wie die SQL-Anweisung CASE oder interne Mechanismen wie referenzielle Integritätsbedingungen oder Auslöser implementiert. Wenn keine Aktion angezeigt wird, werden nur Datenbearbeitungsoperationen verarbeitet, wenn die Bedingung wahr ist.
- Eine der folgenden Angaben wird angezeigt, wenn eine Unterabfrage nach ALL, ANY oder EXISTS im Zugriffsplan verarbeitet wird:
  - `ANY/ALL Subquery`
  - `EXISTS Subquery`
  - `EXISTS SINGLE Subquery`
- Vor bestimmten UPDATE- und DELETE-Operationen muss die Position einer speziellen Zeile in der Tabelle bestimmt werden. Dies wird durch folgende Angabe gekennzeichnet:

  ```
  Establish Row Position
  ```

- Die folgende Angabe wird angezeigt, wenn Zeilen an die Anwendung zurückgegeben werden:

```
Return Data to Application
|  #Columns = n
```

  Wenn die Operation in einen Tabellenzugriff verschoben (Pushdown) wurde, wird eine Abschlussphase (Completion) erforderlich. Diese Phase wird wie folgt angezeigt:

```
Return Data Completion
```

- Die folgenden Informationen werden angezeigt, wenn eine gespeicherte Prozedur aufgerufen wird:

```
Call Stored Procedure
|   Name = schema.funkname
|   Specific Name = spezifischername
|   SQL Access Level = zugriffsebene
|   Language = sprache
|   Parameter Style = parmstil
|   Expected Result Sets = n
|   Fenced                      Not Deterministic
|   Called on NULL Input        Disallow Parallel
|   Not Federated               Not Threadsafe
```

- Die folgenden Informationen werden angezeigt, wenn mindestens ein LOB-Querverweis freigegeben wird:

```
Free LOB Locators
```

Zugehörige Konzepte:

- „Beschreibung der Ausgabe von db2expln und dynexpln" auf Seite 645
- „Beispiele für die Ausgabe von db2expln und dynexpln" auf Seite 665

Beispiele für die Ausgabe von db2expln und dynexpln

Die folgenden Ausgabebeispiele zeigen die EXPLAIN-Informationen, die für bestimmte Abfragen in bestimmten Umgebungen erfasst werden.

Beispiele für die Ausgabe von db2expln und dynexpln

Zum besseren Verständnis finden Sie nachfolgend fünf Beispiele, die den Aufbau und das Format der Ausgabe von `db2expln` und `dynexpln` verdeutlichen. Diese Beispiele wurden für die Beispieldatenbank SAMPLE ausgeführt, die mit DB2® ausgeliefert wird. Jedes Beispiel wird kurz erläutert. Die signifikanten Unterschiede von einem Beispiel zum nächsten wurden **fett** hervorgehoben.

Zugehörige Konzepte:

- „dynexpln" auf Seite 645
- „Beispiel 1: keine Parallelität" auf Seite 666
- „Beispiel 2: Zugriffsplan für Einzelpartition mit partitionsinterner Parallelität" auf Seite 667
- „Beispiel 3: Zugriffsplan für mehrere Partitionen mit partitionsübergreifender Parallelität" auf Seite 669
- „Beispiel 4: Zugriffsplan für mehrere Partitionen mit partitionsübergreifender und partitionsinterner Parallelität" auf Seite 671
- „Beispiel 5: Zugriffsplan für eine zusammengeschlossene Datenbank" auf Seite 674

Zugehörige Referenzen:

• „db2expln - SQL-EXPLAIN-Tool" auf Seite 638

Beispiel 1: keine Parallelität

In diesem Beispiel wird einfach eine Liste aller Namen von Mitarbeitern (employee), ihrer Aufgaben (job), ihrer Abteilungen (department) und Standorte (location) sowie der Namen der Projekte, an denen sie arbeiten, abgerufen. Das wesentliche Merkmal dieses Zugriffsplans besteht darin, dass die relevanten Daten aus allen angegebenen Tabellen durch Hashverknüpfungen verknüpft werden. Da keine Indizes verfügbar sind, führt der Zugriffsplan bei der Verknüpfung einer Tabelle eine Tabellensuche in der jeweiligen Tabelle aus.

```
******************** PACKAGE ****************************************

Package Name = "DOOLE"."EXAMPLE"  Version = ""

 Prep Date = 2002/01/04
 Prep Time = 14:05:00

 Bind Timestamp = 2002-01-04-14.05.00.415403

 Isolation Level          = Cursor Stability
 Blocking                 = Block Unambiguous Cursors
 Query Optimization Class = 5

 Partition Parallel       = No
 Intra-Partition Parallel = No

 SQL Path                 = "SYSIBM", "SYSFUN", "SYSPROC", "DOOLE"

-------------------- SECTION ---------------------------------------
Section = 1


SQL Statement:
  DECLARE EMPCUR CURSOR
  FOR
     SELECT e.lastname, e.job, d.deptname, d.location, p.projname
     FROM employee AS e, department AS d, project AS p
     WHERE e.workdept = d.deptno AND e.workdept = p.deptno


Estimated Cost        = 120.518692
Estimated Cardinality = 221.535980

(    6) Access Table Name = DOOLE.EMPLOYEE  ID = 2,5
         |  #Columns = 3
         |  Relation Scan
         |  |  Prefetch: Eligible
         |  Lock Intents
         |  |  Table: Intent Share
         |  |  Row  : Next Key Share
(    6) |  Process Build Table for Hash Join
(    2) Hash Join
         |  Estimated Build Size: 7111
         |  Estimated Probe Size: 9457
(    5)  |  Access Table Name = DOOLE.PROJECT  ID = 2,7
         |  |  #Columns = 2
         |  |  Relation Scan
         |  |  |  Prefetch: Eligible
         |  |  Lock Intents
         |  |  |  Table: Intent Share
         |  |  |  Row  : Next Key Share
(    5)  |  |  Process Build Table for Hash Join
```

```
(    3) |  Hash Join
        |  |  Estimated Build Size: 5737
        |  |  Estimated Probe Size: 6421
(    4) |  |  Access Table Name = DOOLE.DEPARTMENT  ID = 2,4
        |  |  |  #Columns = 3
        |  |  |  Relation Scan
        |  |  |  |  Prefetch: Eligible
        |  |  |  Lock Intents
        |  |  |  |  Table: Intent Share
        |  |  |  |  Row  : Next Key Share
(    4) |  |  |  Process Probe Table for Hash Join
(    1) Return Data to Application
        |  #Columns = 5

End of section


Optimizer Plan:

              RETURN
              (   1)
                |
HSJOIN
              (   2)
              /      \
        HSJOIN         TBSCAN
        (   3)         (   6)
       /      \          |
   TBSCAN    TBSCAN  Table:
   (   4)    (   5)   DOOLE
     |         |      EMPLOYEE
 Table:      Table:
 DOOLE       DOOLE
 DEPARTMENT  PROJECT
```

Im ersten Teil des Plans wird auf die Tabellen DEPARTMENT und PROJECT zugegriffen, die über eine Hashverknüpfung verknüpft werden. Das Ergebnis dieser Verknüpfung wird mit der Tabelle EMPLOYEE verknüpft. Die Ergebniszeilen werden an die Anwendung zurückgegeben.

Beispiel 2: Zugriffsplan für Einzelpartition mit partitionsinterner Parallelität

In diesem Beispiel wird dieselbe SQL-Anweisung wie im ersten Beispiel gezeigt, jedoch wurde diese Abfrage für eine 4-Wege-SMP-Maschine kompiliert.

```
******************** PACKAGE ****************************************

Package Name = "DOOLE"."EXAMPLE"  Version = ""

 Prep Date = 2002/01/04
 Prep Time = 14:12:38

 Bind Timestamp = 2002-01-04-14.12.38.732627

 Isolation Level          = Cursor Stability
 Blocking                 = Block Unambiguous Cursors
 Query Optimization Class = 5

 Partition Parallel       = No
 Intra-Partition Parallel = Yes (Bind Degree = 4)

 SQL Path                 = "SYSIBM", "SYSFUN", "SYSPROC", "DOOLE"

------------------- SECTION ------------------------------------
```

```
Section = 1


SQL Statement:
  DECLARE EMPCUR CURSOR
  FOR
     SELECT e.lastname, e.job, d.deptname, d.location, p.projname
     FROM employee AS e, department AS d, project AS p
     WHERE e.workdept = d.deptno AND e.workdept = p.deptno

Intra-Partition Parallelism Degree = 4

Estimated Cost        = 133.934692
Estimated Cardinality = 221.535980

(    2) Process Using 4 Subagents
(    7) |  Access Table Name = DOOLE.EMPLOYEE  ID = 2,5
        |  |  #Columns = 3
        |  |  Parallel Scan
        |  |  Relation Scan
        |  |  |  Prefetch: Eligible
        |  |  Lock Intents
        |  |  |  Table: Intent Share
        |  |  |  Row  : Next Key Share
(    7)    |  Process Build Table for Hash Join
(    3)    Hash Join
        |  |  Estimated Build Size: 7111
        |  |  Estimated Probe Size: 9457
(    6)    |  Access Table Name = DOOLE.PROJECT  ID = 2,7
        |  |  |  #Columns = 2
        |  |  |  Parallel Scan
        |  |  |  Relation Scan
        |  |  |  |  Prefetch: Eligible
        |  |  |  Lock Intents
        |  |  |  |  Table: Intent Share
        |  |  |  |  Row  : Next Key Share
(    6)    |  |  Process Build Table for Hash Join
(    4)    |  Hash Join
        |  |  |  Estimated Build Size: 5737
        |  |  |  Estimated Probe Size: 6421
(    5)    |  |  Access Table Name = DOOLE.DEPARTMENT  ID = 2,4
        |  |  |  |  #Columns = 3
        |  |  |  |  Parallel Scan
        |  |  |  |  Relation Scan
        |  |  |  |  |  Prefetch: Eligible
        |  |  |  |  Lock Intents
        |  |  |  |  |  Table: Intent Share
        |  |  |  |  |  Row  : Next Key Share
(    5)    |  |  |  Process Probe Table for Hash Join
(    2)    |  Insert Into Asynchronous Local Table Queue  ID = q1
(    2) Access Local Table Queue  ID = q1  #Columns = 5
(    1) Return Data to Application
        |  #Columns = 5

End of section


Optimizer Plan:

             RETURN
             (   1)
               |
              LTQ
             (   2)
               |
HSJOIN
             (   3)
```

```
             /      \
          HSJOIN          TBSCAN
          (   4)          (   7)
         /      \           |
     TBSCAN     TBSCAN   Table:
     (   5)     (   6)    DOOLE
       |          |       EMPLOYEE
  Table:      Table:
  DOOLE       DOOLE
  DEPARTMENT  PROJECT
```

Dieser Plan stimmt mit dem Plan des ersten Beispiels weitgehend überein. Der Unterschied besteht in der Erstellung von vier Subagenten beim ersten Starten des Plans und der Tabellenwarteschlange am Ende des Plans, um die Ergebnisse der Arbeit aller Subagenten aufzunehmen, bevor sie an die Anwendung zurückgegeben werden.

Beispiel 3: Zugriffsplan für mehrere Partitionen mit partitionsübergreifender Parallelität

Dieses Beispiel zeigt wiederum dieselbe SQL-Anweisung wie das erste Beispiel, jedoch wurde hier die Abfrage auf einer partitionierten Datenbank kompiliert, die aus drei Datenbankpartitionen besteht.

```
******************** PACKAGE ****************************************

Package Name = "DOOLE"."EXAMPLE"  Version = ""

 Prep Date = 2002/01/04
 Prep Time = 14:54:57

 Bind Timestamp = 2002-01-04-14.54.57.033666

 Isolation Level          = Cursor Stability
 Blocking                 = Block Unambiguous Cursors
 Query Optimization Class = 5

 Partition Parallel       = Yes
 Intra-Partition Parallel = No

 SQL Path                 = "SYSIBM", "SYSFUN", "SYSPROC", "DOOLE"

-------------------- SECTION ---------------------------------------
Section = 1


SQL Statement:
  DECLARE EMPCUR CURSOR
  FOR
     SELECT e.lastname, e.job, d.deptname, d.location, p.projname
     FROM employee AS e, department AS d, project AS p
     WHERE e.workdept = d.deptno AND e.workdept = p.deptno


Estimated Cost        = 118.483406
Estimated Cardinality = 474.720032

Coordinator Subsection:(-----)    Distribute Subsection #2
           |  Broadcast to Node List
           |  |  Nodes = 10, 33, 55
(-----)    Distribute Subsection #3
           |  Broadcast to Node List
           |  |  Nodes = 10, 33, 55
(-----)    Distribute Subsection #1
```

```
            |  Broadcast to Node List
            |  |  Nodes = 10, 33, 55
(    2)     Access Table Queue  ID = q1  #Columns = 5
(    1)     Return Data to Application
            |  #Columns = 5

Subsection #1:(    8)    Access Table Queue  ID = q2  #Columns = 2
(    3)     Hash Join
            |  Estimated Build Size: 5737
            |  Estimated Probe Size: 8015
(    6)     |  Access Table Queue  ID = q3  #Columns = 3
(    4)     |  Hash Join
            |  |  Estimated Build Size: 5333
            |  |  Estimated Probe Size: 6421
(    5)     |  |  Access Table Name = DOOLE.DEPARTMENT  ID = 2,4
            |  |  |  #Columns = 3
            |  |  |  Relation Scan
            |  |  |  |  Prefetch: Eligible
            |  |  |  Lock Intents
            |  |  |  |  Table: Intent Share
            |  |  |  |  Row  : Next Key Share
(    5)     |  |  |  Process Probe Table for Hash Join
(    2)     Insert Into Asynchronous Table Queue  ID = q1
            |  Broadcast to Coordinator Node
            |  Rows Can Overflow to Temporary Table

         Subsection #2:
(    9)     Access Table Name = DOOLE.PROJECT  ID = 2,7
            |  #Columns = 2
            |  Relation Scan
            |  |  Prefetch: Eligible
            |  Lock Intents
            |  |  Table: Intent Share
            |  |  Row  : Next Key Share
(    9)     |  Insert Into Asynchronous Table Queue  ID = q2
            |  |  Hash to Specific Node
            |  |  Rows Can Overflow to Temporary Tables
(    8)     Insert Into Asynchronous Table Queue Completion  ID = q2

         Subsection #3:
(    7)     Access Table Name = DOOLE.EMPLOYEE  ID = 2,5
            |  #Columns = 3
            |  Relation Scan
            |  |  Prefetch: Eligible
            |  Lock Intents
            |  |  Table: Intent Share
            |  |  Row  : Next Key Share
(    7)     |  Insert Into Asynchronous Table Queue  ID = q3
            |  |  Hash to Specific Node
            |  |  Rows Can Overflow to Temporary Tables
(    6)     Insert Into Asynchronous Table Queue Completion  ID = q3

End of section


Optimizer Plan:

              RETURN
              (   1)
                |
               BTQ
              (   2)
                |
HSJOIN
              (   3)
              /     \
         HSJOIN          DTQ
```

```
        (   4)          (   8)
       /      \           |
    TBSCAN      DTQ      TBSCAN
    (   5)     (   6)    (   9)
      |          |        |
 Table:        TBSCAN    Table:
 DOOLE         (   7)    DOOLE
 DEPARTMENT      |       PROJECT
 Table:
 DOOLE
 EMPLOYEE
```

Dieser Plan enthält dieselben Bestandteile wie der Plan im ersten Beispiel, aber der Bereich wurde in vier Teilbereiche (Subsection) unterteilt. Die Teilbereiche haben folgende Aufgaben:

- **Coordinator Subsection (Koordinatorteilbereich)**. Dieser Teilbereich koordiniert die anderen Teilbereiche. In diesem Plan sorgt er dafür, dass die anderen Teilbereiche verteilt werden, und er verwendet anschließend eine Tabellenwarteschlange, um die Ergebnisse zu sammeln, die an die Anwendung zurückgegeben werden sollen.
- **Subsection #1**. Dieser Teilbereich durchsucht die Tabellenwarteschlange q2 und verwendet eine Hashverknüpfung, um sie mit den Daten aus Tabellenwarteschlange q3 zu verknüpfen. Eine zweite Mischverknüpfung fügt anschließend die Daten aus der Tabelle DEPARTMENT hinzu. Die verknüpften Zeilen werden dann über Tabellenwarteschlange q1 an den Koordinatorteilbereich gesendet.
- **Subsection #2**. Dieser Teilbereich durchsucht die Tabelle PROJECT und verteilt sie per Hashverfahren mit den Ergebnissen an einen bestimmten Knoten. Diese Ergebnisse werden von Subsection #1 gelesen.
- **Subsection #3**. Dieser Teilbereich durchsucht die Tabelle EMPLOYEE und verteilt sie per Hashverfahren mit den Ergebnissen an einen bestimmten Knoten. Diese Ergebnisse werden von Subsection #1 gelesen.

Beispiel 4: Zugriffsplan für mehrere Partitionen mit partitionsübergreifender und partitionsinterner Parallelität

In diesem Beispiel wird dieselbe SQL-Anweisung wie im ersten Beispiel gezeigt, jedoch wurde die Abfrage auf einer partitionierten Datenbank mit drei Datenbankpartitionen kompiliert, die sich jeweils auf einer 4-Wege-SMP-Maschine befinden.

```
******************** PACKAGE ************************************

Package Name = "DOOLE"."EXAMPLE"  Version = ""

 Prep Date = 2002/01/04
 Prep Time = 14:58:35

 Bind Timestamp = 2002-01-04-14.58.35.169555

 Isolation Level          = Cursor Stability
 Blocking                 = Block Unambiguous Cursors
 Query Optimization Class = 5

 Partition Parallel       = Yes
 Intra-Partition Parallel = Yes (Bind Degree = 4)

 SQL Path                 = "SYSIBM", "SYSFUN", "SYSPROC", "DOOLE"

-------------------- SECTION -------------------------------------
Section = 1
```

```
SQL Statement:
  DECLARE EMPCUR CURSOR
  FOR
     SELECT e.lastname, e.job, d.deptname, d.location, p.projname
     FROM employee AS e, department AS d, project AS p
     WHERE e.workdept = d.deptno AND e.workdept = p.deptno

Intra-Partition Parallelism Degree = 4

Estimated Cost        = 145.198898
Estimated Cardinality = 474.720032

Coordinator Subsection:(-----)    Distribute Subsection #2
            |  Broadcast to Node List
            |  |  Nodes = 10, 33, 55
(-----)    Distribute Subsection #3
            |  Broadcast to Node List
            |  |  Nodes = 10, 33, 55
(-----)    Distribute Subsection #1
            |  Broadcast to Node List
            |  |  Nodes = 10, 33, 55
(    2)    Access Table Queue  ID = q1  #Columns = 5
(    1)    Return Data to Application
            |  #Columns = 5

Subsection #1:(    3)    Process Using 4 Subagents
(   10)     |  Access Table Queue  ID = q3  #Columns = 2
(    4)     |  Hash Join
            |  |  Estimated Build Size: 5737
            |  |  Estimated Probe Size: 8015
(    7)     |  |  Access Table Queue  ID = q5  #Columns = 3
(    5)     |  |  Hash Join
            |  |  |  Estimated Build Size: 5333
            |  |  |  Estimated Probe Size: 6421
(    6)     |  |  |  Access Table Name = DOOLE.DEPARTMENT  ID = 2,4
            |  |  |  |  #Columns = 3
            |  |  |  |  Parallel Scan
            |  |  |  |  Relation Scan
            |  |  |  |  |  Prefetch: Eligible
            |  |  |  |  Lock Intents
            |  |  |  |  |  Table: Intent Share
            |  |  |  |  |  Row  : Next Key Share
(    6)     |  |  |  |  Process Probe Table for Hash Join
(    3)     |  Insert Into Asynchronous Local Table Queue  ID = q2
(    3)    Access Local Table Queue  ID = q2  #Columns = 5
(    2)    Insert Into Asynchronous Table Queue  ID = q1
            |  Broadcast to Coordinator Node
            |  Rows Can Overflow to Temporary Table

        Subsection #2:
(   11)    Process Using 4 Subagents
(   12)     |  Access Table Name = DOOLE.PROJECT  ID = 2,7
            |  |  #Columns = 2
            |  |  Parallel Scan
            |  |  Relation Scan
            |  |  |  Prefetch: Eligible
            |  |  Lock Intents
            |  |  |  Table: Intent Share
            |  |  |  Row  : Next Key Share
(   11)     |  Insert Into Asynchronous Local Table Queue  ID = q4
(   11)    Access Local Table Queue  ID = q4  #Columns = 2
(   10)    Insert Into Asynchronous Table Queue  ID = q3
            |  Hash to Specific Node
            |  Rows Can Overflow to Temporary Tables
```

```
        Subsection #3:
(    8)    Process Using 4 Subagents
(    9)    |  Access Table Name = DOOLE.EMPLOYEE  ID = 2,5
           |  |  #Columns = 3
           |  |  Parallel Scan
           |  |  Relation Scan
           |  |  |  Prefetch: Eligible
           |  |  Lock Intents
           |  |  |  Table: Intent Share
           |  |  |  Row  : Next Key Share
(    8)    |  Insert Into Asynchronous Local Table Queue  ID = q6
(    8)    Access Local Table Queue  ID = q6  #Columns = 3
(    7)    Insert Into Asynchronous Table Queue  ID = q5
           |  Hash to Specific Node
           |  Rows Can Overflow to Temporary Tables

End of section


Optimizer Plan:

             RETURN
             (   1)
               |
              BTQ
             (   2)
               |
              LTQ
             (   3)
               |
HSJOIN
             (   4)
             /     \
        HSJOIN      DTQ
        (   5)     (  10)
       /     \       |
   TBSCAN    DTQ     LTQ
   (   6)   (   7)  (  11)
     |        |      |
 Table:      LTQ    TBSCAN
 DOOLE      (   8)  (  12)
 DEPARTMENT   |      |
  TBSCAN            Table:
            (   9)  DOOLE
              |     PROJECT
 Table:
 DOOLE
 EMPLOYEE
```

Dieser Plan ist ähnlich wie der im dritten Beispiel, nur dass mehrere Subagenten jeden Teilbereich ausführen. Außerdem sammelt am Ende eines jeden Teilbereichs eine lokale Tabelle die Ergebnisse von allen Subagenten, bevor die entsprechenden Zeilen in die zweite Tabellenwarteschlange eingefügt werden, die per Hashverfahren an einen bestimmten Knoten gesendet wird.

Beispiel 5: Zugriffsplan für eine zusammengeschlossene Datenbank

In diesem Beispiel wird dieselbe SQL-Anweisung wie im ersten Beispiel gezeigt. Die Abfrage wurde jedoch auf einer zusammengeschlossenen Datenbank kompiliert, bei der sich die Tabellen DEPARTMENT und PROJECT auf einer Datenquelle befinden und die Tabelle EMPLOYEE auf dem Server für die zusammengeschlossenen Datenbanken ist.

```
******************** PACKAGE ************************************

Package Name = "DOOLE"."EXAMPLE"  Version = ""

 Prep Date = 2002/01/11
 Prep Time = 13:52:48

 Bind Timestamp = 2002-01-11-13.52.48.325413

 Isolation Level          = Cursor Stability
 Blocking                 = Block Unambiguous Cursors
 Query Optimization Class = 5

 Partition Parallel       = No
 Intra-Partition Parallel = No

 SQL Path                 = "SYSIBM", "SYSFUN", "SYSPROC", "DOOLE"

-------------------- SECTION ---------------------------------------
Section = 1


SQL Statement:
  DECLARE EMPCUR CURSOR
  FOR
     SELECT e.lastname, e.job, d.deptname, d.location, p.projname
     FROM employee AS e, department AS d, project AS p
     WHERE e.workdept = d.deptno AND e.workdept = p.deptno


Estimated Cost        = 1804.625000
Estimated Cardinality = 112000.000000

(    7) Ship Distributed Subquery #2
        |  #Columns = 2
(    2) Hash Join
        |  Estimated Build Size: 48444
        |  Estimated Probe Size: 232571
(    6) |  Access Table Name = DOOLE.EMPLOYEE  ID = 2,5
        |  |  #Columns = 3
        |  |  Relation Scan
        |  |  |  Prefetch: Eligible
        |  |  Lock Intents
        |  |  |  Table: Intent Share
        |  |  |  Row  : Next Key Share
(    6) |  |  Process Build Table for Hash Join
(    3) |  Hash Join
        |  |  Estimated Build Size: 7111
        |  |  Estimated Probe Size: 64606
(    4) |  |  Ship Distributed Subquery #1
        |  |  |  #Columns = 3
(    1) Return Data to Application
        |  #Columns = 5
```

```
Distributed Substatement #1:
(    4) Server: REMOTE  (DB2/UDB 8.1)
SQL Statement:

        SELECT A0."DEPTNO", A0."DEPTNAME", A0."LOCATION"
        FROM "DOOLE"."DEPARTMENT" A0

   Nicknames Referenced:
           DOOLE.DEPARTMENT  ID = 32768
              Base = DOOLE.DEPARTMENT
        #Output Columns = 3



Distributed Substatement #2:
(    7) Server: REMOTE  (DB2/UDB 8.1)
SQL Statement:

        SELECT A0."DEPTNO", A0."PROJNAME"
        FROM "DOOLE"."PROJECT" A0

   Nicknames Referenced:
           DOOLE.PROJECT  ID = 32769
               Base = DOOLE.PROJECT
        #Output Columns = 2

End of section

Optimizer Plan:

             RETURN
             (   1)
                |
HSJOIN
             (   2)
              /     \
        HSJOIN          SHIP
        (   3)         (   7)
       /     \           |
    SHIP     TBSCAN   Nickname:
   (   4)    (   6)   DOOLE
     |         |      PROJECT
 Nickname:    Table: DOOLE       DOOLE
 DEPARTMENT   EMPLOYEE
```

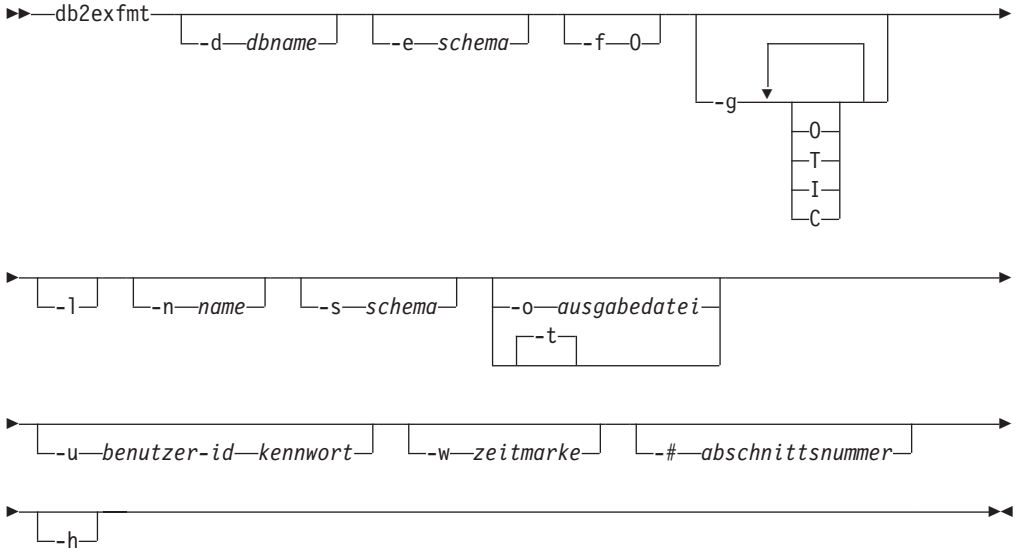
Dieser Plan enthält dieselben Bestandteile wie der Plan im ersten Beispiel. Ausgenommen hiervon sind die Daten für zwei der Tabellen, die von Datenquellen kommen. Auf diese beiden Tabellen wird über verteilte Unterabfragen zugegriffen, die in diesem Fall einfach alle Zeilen der betreffenden Tabellen auswählen. Wenn die Daten zum Server mit zusammengeschlossenen Datenbanken zurückgegeben werden, werden sie mit den Daten aus der lokalen Tabelle verknüpft.

Anhang D. db2exfmt - Formatierungstool für EXPLAIN-Tabellen

Mit dem Tool `db2exfmt` können Sie den Inhalt der EXPLAIN-Tabellen formatieren. Dieses Tool befindet sich im Unterverzeichnis `misc` des Verzeichnisses `sqllib` für Ihr Exemplar.

Sie müssen über Lesezugriff auf die zu formatierenden EXPLAIN-Tabellen verfügen, um dieses Tool verwenden zu können.

Befehlssyntax:



Befehlsparameter:

-d *dbname*
Name der Datenbank mit den Paketen

-e *schema*
EXPLAIN-Tabellenschema

-f Formatierungsmarkierungen. In diesem Release wird nur der Wert `O` (Operatorzusammenfassung) unterstützt.

-g Diagrammzugriffsplan. Wenn nur `-g` angegeben wird, wird ein Diagramm gefolgt von den Formatierungsangaben für alle Tabellen generiert. Ansonsten kann eine beliebige Kombination der folgenden gültigen Werte angegeben werden:

- **O** Nur ein Diagramm wird generiert. Der Tabelleninhalt wird nicht formatiert.
- **T** In das Diagramm wird der Gesamtaufwand unter jedem Operator aufgenommen.
- **I** In das Diagramm wird der Ein-/Ausgabeaufwand unter jedem Operator aufgenommen.
- **C** In das Diagramm wird die erwartete Ausgabekardinalität (Anzahl Tupel) für jeden Operator aufgenommen.

-l Die Groß-/Kleinschreibung wird während der Verarbeitung der Paketnamen beachtet.

-n *name*
Name für die Quelle der EXPLAIN-Anforderung (SOURCE_NAME)

-s *schema*
Schema oder Qualifikationsmerkmal für die Quelle der EXPLAIN-Anforderung (SOURCE_SCHEMA)

-o *ausgabedatei*
Name der Ausgabedatei

-t Leiten der Ausgabe an die Workstation

-u *benutzer-id kennwort*
Mit dieser Option werden eine Benutzer-ID und ein Kennwort für die Verbindung zu einer Datenbank angegeben.

Sowohl die Benutzer-ID als auch das Kennwort müssen gemäß den Namenskonventionen gültig sein und von der Datenbank erkannt werden.

-w *zeitmarke*
EXPLAIN-Zeitmarke. Geben Sie `-1` an, um die letzte EXPLAIN-Anforderung abzurufen.

-# *abschnittsnummer*
Abschnittsnummer in der Quelle. Geben Sie null an, um alle Abschnitte anzufordern.

-h Anzeigen des Hilfetexts. Wenn diese Option angegeben wird, werden alle anderen Optionen ignoriert, und lediglich der Hilfetext wird angezeigt.

Hinweise zur Verwendung:

Sie werden aufgefordert, die nicht angegebenen Parameterwerte anzugeben, oder Ihnen wird mitgeteilt, dass nicht alle angegeben wurden, ausgenommen im Fall der Optionen `-h` und `-l`.

Wenn kein EXPLAIN-Tabellenschema angegeben wird, wird der Wert der Umgebungsvariable **USER** als Standardwert verwendet. Wenn diese Variable nicht gefunden wird, wird der Benutzer aufgefordert, ein EXPLAIN-Tabellenschema anzugeben.

Der Quellenname, das Quellenschema und die EXPLAIN-Zeitmarke können in Form eines LIKE-Vergleichselements angegeben werden, in dem das Prozentzeichen (%) und das Unterstreichungszeichen (_) als Platzhalterzeichen zur Auswahl mehrerer Quellen bei einem Aufruf verwendet werden können. Für die letzte mit EXPLAIN bearbeitete Anweisung kann die EXPLAIN-Zeit als `-1` angegeben werden. Wenn `-o` ohne Dateiname und `-t` nicht angegeben wird, wird der Benutzer aufgefordert, einen Dateinamen anzugeben (der Standardname ist `db2exfmt.out`). Wenn weder `-o` noch `-t` angegeben wird, wird der Benutzer aufgefordert, einen Dateinamen anzugeben (die Standardoption ist Terminalausgabe). Wenn sowohl `-o` als auch `-t` angegeben werden, wird die Ausgabe an das Terminal geleitet.

Zugehörige Konzepte:

- „EXPLAIN-Tools" auf Seite 224
- „Richtlinien zur Verwendung von EXPLAIN-Informationen" auf Seite 226
- „Richtlinien zur Erfassung von EXPLAIN-Informationen" auf Seite 234

Anhang E. Knotenübergreifende Fehlerbehebung mit dem Befehl 'db2adutl' und den Datenbankkonfigurationsparametern 'logarchopt1' und 'vendoropt'

Die folgenden Beispiele zeigen, wie eine knotenübergreifende Fehlerbehebung mit dem Befehl 'db2adutl', sowie wie mit Hilfe der Datenbankkonfigurationsparameter *logarchopt1* und *vendoropt* ausgeführt wird. In den folgenden Beispielen besitzt Computer 1 den Namen `bar` und wird unter AIX betrieben. Der Eigner dieser Maschine ist der Benutzer `roecken`. Die Datenbank auf der Maschine `bar` heißt `zample`. Computer 2 besitzt den Namen `dps`. Diese Maschine wird ebenfalls unter AIX betrieben und ihr Eigner ist `regress9`.

PASSWORDACCESS = Generate:

Computer 1:

1. Richten Sie die Datenbank zur Protokollarchivierung in TSM ein. Aktualisieren Sie den Datenbankkonfigurationsparameter *logarchmeth1* für die Datenbank `zample`:

   ```
   bar:/home/roecken> db2 update db cfg for zample using LOGARCHMETH1 tsm
   ```

   Die folgenden Informationen werden zurückgegeben:

   ```
   DB20000I  Der Befehl UPDATE DATABASE CONFIGURATION wurde erfolgreich
   ausgeführt.
   ```

   Anmerkung: Vor der Aktualisierung der Datenbankkonfiguration kann es erforderlich sein, eine Offlinesicherung der Datenbank zu erstellen.

2. Erstellen Sie eine Onlinesicherung der Datenbank:

   ```
   db2 backup db zample online use tsm
   ```

   Die folgenden Informationen werden zurückgegeben:

   ```
   Sicherung erfolgreich. Die Zeitmarke für dieses Sicherungsimage ist:
   20040216151025
   ```

3. Stellen Sie eine Verbindung ('Connect') zur Datenbank `zample` her und erstellen Sie eine Tabelle in der Datenbank.

4. Laden Sie Daten in die neue Tabelle. In diesem Beispiel hat die Tabelle den Namen `a`, und die Daten werden aus einer Datei mit ASCII-Format mit Begrenzern mit dem Namen `mr` geladen. Die Option COPY YES wird angegeben, um eine Kopie der geladenen Daten zu erstellen, und durch die Option USE TSM wird angegeben, dass die Kopie der Daten in Tivoli Storage Manager zu speichern ist.

   Anmerkung: Sie können die Option COPY YES nur angeben, wenn die Datenbank für die aktualisierende Wiederherstellung konfiguriert ist. Das heißt, der Datenbankkonfigurationsparameter *logretain* oder *userexit* (oder beide) müssen für die Datenbank aktiviert sein.

   ```
   bar:/home/roecken> db2 load from mr of del modified by noheader replace
   into a copy yes use tsm
   ```

   Das Dienstprogramm gibt eine Reihe von Nachrichten zurück, um den Verarbeitungsfortschritt anzuzeigen:

   ```
   SQL3109N  Das Dienstprogramm beginnt mit dem Laden von Daten aus der Datei
   "/home/roecken/mr".
   ```

```
SQL3500W  Die Phase "LOAD" wird gestartet (Zeit: "02/16/2004
15:12:13.392633").

SQL3519W  Synchronisationspunkt am Beginn des Ladevorgangs. Eingabesatzzähler = "0".

SQL3520W  Synchronisationspunkt für Ladevorgang erfolgreich.

SQL3110N  Die Verarbeitung durch das Dienstprogramm ist abgeschlossen.
          "1" Zeilen aus der Eingabedatei wurden gelesen.

SQL3519W  Synchronisationspunkt am Beginn des Ladevorgangs.
          Eingabesatzzähler = "1".

SQL3520W  Synchronisationspunkt für Ladevorgang erfolgreich.

SQL3515W  Die Phase "LOAD" wurde beendet
          (Zeit: "02/16/2004  15:12:13.445718").

Anzahl gelesener Zeilen          = 1
Anzahl übersprungener Zeilen     = 0
Anzahl geladener Zeilen          = 1
Anzahl zurückgewiesener Zeilen   = 0
Anzahl gelöschter Zeilen         = 0
Anzahl festgeschriebener Zeilen  = 1
```

Nun sollte ein Sicherungsimage, ein Ladekopieimage und eine Protokolldatei in TSM vorhanden sein. Eine Abfrage auf die Datenbank `zample` kann wie folgt ausgeführt werden:

```
bar:/home/roecken> db2adutl query db zample
```

Die folgenden Informationen werden zurückgegeben:

```
Retrieving FULL DATABASE BACKUP information.
    1 Time: 20040216151025  Oldest log: S0000000.LOG  DB Partition Number: 0
    Sessions: 1


Retrieving INCREMENTAL DATABASE BACKUP information.
  No INCREMENTAL DATABASE BACKUP images found for ZAMPLE


Retrieving DELTA DATABASE BACKUP information.
  No DELTA DATABASE BACKUP images found for ZAMPLE


Retrieving TABLESPACE BACKUP information.
  No TABLESPACE BACKUP images found for ZAMPLE


Retrieving INCREMENTAL TABLESPACE BACKUP information.
  No INCREMENTAL TABLESPACE BACKUP images found for ZAMPLE


Retrieving DELTA TABLESPACE BACKUP information.
  No DELTA TABLESPACE BACKUP images found for ZAMPLE


Retrieving LOAD COPY information.
    1 Time: 20040216151213


Retrieving LOG ARCHIVE information.
   Log file: S0000000.LOG, Chain Num: 0, DB Partition Number: 0,
   Taken at: 2004-02-16-15.10.38
```

5. Zur Aktivierung der knotenübergreifenden Fehlerbehebung muss einem anderen Knoten und einem anderen Konto Zugriff auf die Objekte auf dem Computer bar erteilt werden. In diesem Beispiel wird der Zugriff dem Knoten dps und dem Benutzer regress9 erteilt.

   ```
   bar:/home/roecken> db2adutl grant user regress9 on nodename dps for db zample
   ```

   Die folgenden Informationen werden zurückgegeben:

   ```
   Successfully added permissions for regress9 to access ZAMPLE on node dps.
   ```

   Zur Abfrage der Ergebnisse der **db2adutl grant**-Operation führen Sie den folgenden Befehl aus:

   ```
   bar:/home/roecken> db2adutl queryaccess
   ```

   Die folgenden Informationen werden zurückgegeben:

   ```
   Node                Username            Database Name   Type
   ------------------------------------------------------------
   DPS                 regress9            ZAMPLE          A
   ------------------------------------------------------------
    Access Types:  B - backup images   L - logs   A - both
   ```

PASSWORDACCESS = Generate-Umgebung:

Computer 2:

Computer 2 (dps) ist noch nicht konfiguriert. Eine **db2adutl**-Abfrage auf dps für die Datenbank zample liefert die folgenden Ergebnisse:

```
dps:/home/regress9> db2adutl query db zample
--- Database directory is empty ---
Warning: There are no file spaces created by DB2 on the ADSM server
Warning: No DB2 backup images found in ADSM for any alias.


dps:/home/regress9> db2adutl query db zample nodename bar owner roecken
--- Database directory is empty ---

Query for database ZAMPLE


Retrieving FULL DATABASE BACKUP information.
    1 Time: 20040216151025  Oldest log: S0000000.LOG  DB Partition Number: 0
    Sessions: 1


Retrieving INCREMENTAL DATABASE BACKUP information.
  No INCREMENTAL DATABASE BACKUP images found for ZAMPLE


Retrieving DELTA DATABASE BACKUP information.
  No DELTA DATABASE BACKUP images found for ZAMPLE


Retrieving TABLESPACE BACKUP information.
  No TABLESPACE BACKUP images found for ZAMPLE


Retrieving INCREMENTAL TABLESPACE BACKUP information.
  No INCREMENTAL TABLESPACE BACKUP images found for ZAMPLE


Retrieving DELTA TABLESPACE BACKUP information.
  No DELTA TABLESPACE BACKUP images found for ZAMPLE


Retrieving LOAD COPY information.
```

```
     1 Time: 20040216151213


Retrieving LOG ARCHIVE information.
   Log file: S0000000.LOG, Chain Num: 0, DB Partition Number: 0,
   Taken at: 2004-02-16-15.10.38
```

Die Datenbank zample ist auf dem Computer dps noch nicht vorhanden.

1. Stellen Sie die Datenbank zample auf dem Computer dps wieder her:

   ```
   dps:/home/regress9> db2 restore db zample use tsm options
   "'-fromnode=bar -fromowner=roecken'" without prompting
   ```

   Die folgenden Informationen werden zurückgegeben:

   ```
   DB20000I  Der Befehl RESTORE DATABASE wurde erfolgreich ausgeführt.
   ```

   Anmerkung: Wenn die Datenbank zample auf dps bereits vorhanden wäre, würde der Parameter OPTIONS weggelassen, und der Datenbankkonfigurationsparameter *vendoropt* würde verwendet. Dieser Konfigurationsparameter überschreibt den Parameter OPTIONS für eine BACKUP- oder RESTORE-Operation.

   Eine aktualisierende Wiederherstellung (ROLLFORWARD) in der Datenbank zample würde fehlschlagen, weil das Dienstprogramm zur aktualisierenden Wiederherstellung die Protokolldateien nicht finden kann. Eine ROLLFORWARD-Operation wie die folgende:

   ```
   dps:/home/regress9> db2 rollforward db zample to end of logs and stop
   ```

   liefert den folgenden Fehler:

   ```
   SQL4970N  Die aktualisierende Wiederherstellung der Datenbank "ZAMPLE"
   kann wegen fehlender Protokolldatei(en) auf Knoten "0" nicht den angegebenen
   Endpunkt (Protokollende oder angegebener Zeitpunkt) erreichen.
   ```

2. Um das Dienstprogramm ROLLFORWARD zu veranlassen, auf einer anderen Maschine nach den Protokolldateien zu suchen, müssen Sie den richtigen *logarchopt*-Wert konfigurieren, in diesem Fall den Datenbankkonfigurationsparameter *logarchopt1*:

   ```
   dps:/home/regress9> db2 update db cfg for zample using logarchopt1
   "'-fromnode=bar -fromowner=roecken'"
   ```

3. Damit das Dienstprogramm ROLLFORWARD die Ladekopieimages verwenden kann, müssen Sie auch den Datenbankkonfigurationsparameter *vendoropt* definieren:

   ```
   dps:/home/regress9> db2 update db cfg for zample using VENDOROPT
   "'-fromnode=bar -fromowner=roecken'"
   ```

4. Die Datenbank zample kann nun aktualisierend wiederhergestellt werden:

   ```
   dps:/home/regress9> db2 rollforward db zample to end of logs and stop
   ```

   Die folgenden Informationen werden zurückgegeben:

   ```
                                  Status der aktualisierenden Wiederherstellung

   Aliasname der Eingabedatenbank             = zample
   Anzahl der Knoten, die Status zurückgaben  = 1

   Knotennummer                               = 0
   Aktualisierende Wiederherstellung: Status  = nicht anstehend
   Nächste zu lesende Protokolldatei          =
   Verarbeitete Protokolldateien              = S0000000.LOG - S0000000.LOG
   Letzte festgeschriebene Transaktion        = 2004-02-16-20.10.38.000000

  DB20000I  Der Befehl ROLLFORWARD wurde erfolgreich ausgeführt.
   ```

PASSWORDACCESS = Prompt-Umgebung:

In einer PROMPT-Umgebung sind zusätliche Informationen erforderlich. Dies sind insbesondere der TSM-Knotenname und das Kennwort der Maschine, auf der die Objekte erstellt wurden.

Aktualisieren Sie für **db2adutl** die Datei `dsm.sys` (auf Windows-basierten Plattformen die Datei *dsm.opt*file) und fügen Sie NODENAME `bar` der Serverklausel hinzu (da `bar` der Name des Quellensystems ist):

```
dps:/home/regress9> db2adutl query db zample nodename bar
owner roecken password *******
```

Die folgenden Informationen werden zurückgegeben:

```
Query for database ZAMPLE

Retrieving FULL DATABASE BACKUP information.
    1 Time: 20040216151025  Oldest log: S0000000.LOG  DB Partition Number: 0
    Sessions: 1

Retrieving INCREMENTAL DATABASE BACKUP information.
  No INCREMENTAL DATABASE BACKUP images found for ZAMPLE

Retrieving DELTA DATABASE BACKUP information.
  No DELTA DATABASE BACKUP images found for ZAMPLE

Retrieving TABLESPACE BACKUP information.
  No TABLESPACE BACKUP images found for ZAMPLE

Retrieving INCREMENTAL TABLESPACE BACKUP information.
  No INCREMENTAL TABLESPACE BACKUP images found for ZAMPLE

Retrieving DELTA TABLESPACE BACKUP information.
  No DELTA TABLESPACE BACKUP images found for ZAMPLE

Retrieving LOAD COPY information.
    1 Time: 20040216151213

Retrieving LOG ARCHIVE information.
   Log file: S0000000.LOG, Chain Num: 0, DB Partition Number: 0,
   Taken at: 2004-02-16-15.10.38
```

1. Wenn die Datenbank nicht vorhanden ist, erstellen Sie eine leere Datenbank `zample`. Ist die Datenbank `zample` bereits vorhanden, können dieser und die nächsten beiden Schritte, in denen die Datenbankkonfiguration aktualisiert wird, übersprungen werden.

   ```
   dps:/home/regress9> db2 create db zample
   ```

2. Aktualisieren Sie den Datenbankkonfigurationsparameter *tsm_nodename* für die Datenbank `zample`:

   ```
   dps:/home/regress9> db2 update db cfg for zample using tsm_nodename bar
   ```

3. Aktualisieren Sie den Datenbankkonfigurationsparameter *tsm_password* für die Datenbank `zample`:

   ```
   dps:/home/regress9> db2 update db cfg for zample using
   tsm_password ********
   ```

4. Stellen Sie die Datenbank `zample` wieder her:

   ```
   dps:/home/regress9> db2 restore db zample use tsm options
   "'-fromnode=bar -fromowner=roecken'" without prompting
   ```

Die Wiederherstellungsoperation wird erfolgreich ausgeführt, jedoch wird eine Warnung ausgeben:

```
SQL2540W  RESTORE wurde erfolgreich ausgeführt, es wurde jedoch eine
Warnung "2523" bei der Wiederherstellung der Datenbank im Modus
'No Interrupt' festgestellt.
```

An diesem Punkt kann das Dienstprogramm zur aktualisierenden Wiederherstellung wiederum die richtigen Protokolldateien nicht finden:

```
dps:/home/regress9> db2 rollforward db zample to end of logs and stop
```

Die folgende Fehlernachricht wird zurückgegeben:

```
SQL1268N  Die aktualisierende Wiederherstellung wurde nach dem
Fehler "-2112880618" beendet, während die Protokolldatei "S0000000.LOG"
für die Datenbank "ZAMPLE" auf dem Knoten "0" abgerufen wurde.
```

5. Da die Operation der Datenbankwiederherstellung die Datenbankkonfigurationsdatei ersetzt, müssen die TSM-Datenbankkonfigurationswerte auf die korrekten Werte gesetzt werden. Zunächst muss der Konfigurationsparameter *tsm_nodename* zurückgesetzt werden:

   ```
   dps:/home/regress9> db2 update db cfg for zample using tsm_nodename bar
   ```

6. Der Datenbankkonfigurationsparameter *tsm_password* muss zurückgesetzt werden:

   ```
   dps:/home/regress9> db2 update db cfg for zample using tsm_password *******
   ```

7. Der Datenbankkonfigurationsparameter *logarchopt1* muss zurückgesetzt werden, so dass vom Dienstprogramm ROLLFORWARD die richtigen Protokolldateien gefunden werden:

   ```
   dps:/home/regress9> db2 update db cfg for zample using logarchopt1
   "'-fromnode=bar -fromowner=roecken'"
   ```

8. Der Datenbankkonfigurationsparameter *vendoropt* muss ebenfalls zurückgesetzt werden, so dass auch die Wiederherstellungsdatei verwendet werden kann:

   ```
   dps:/home/regress9> db2 update db cfg for zample using VENDOROPT
   "'-fromnode=bar -fromowner=roecken'"
   ```

9. Wenn die Datenbankkonfigurationsparameter zurückgesetzt wurden, kann die Datenbank aktualisierend wiederhergestellt werden:

   ```
   dps:/home/regress9> db2 rollforward db zample to end of logs and stop
   ```

   Der Befehl ROLLFORWARD QUERY STATUS für die Datenbank `zample` zeigt Folgendes:

   ```
                                  Status der aktualisierenden Wiederherstellung

    Aliasname der Eingabedatenbank              = zample
    Anzahl der Knoten, die Status zurückgaben   = 1

    Knotennummer                                = 0
    Aktualisierende Wiederherstellung: Status   = nicht anstehend
    Nächste zu lesende Protokolldatei           =
    Verarbeitete Protokolldateien               = S0000000.LOG - S0000000.LOG
    Letzte festgeschriebene Transaktion         = 2004-02-16-20.10.38.000000

   DB20000I  Der Befehl ROLLFORWARD wurde erfolgreich ausgeführt.
   ```

Zugehörige Referenzen:

- „db2adutl - Managing DB2 objects within TSMCommand" in *Command Reference*
- „logarchopt1 - Optionen für primäres Protokollarchiv" auf Seite 470
- „vendoropt - Lieferantenoptionen" auf Seite 476

Anhang F. Technische Informationen zu DB2 Universal Database

DB2-Dokumentation und Hilfe

Die technischen Informationen zu DB2® stehen über die folgenden Tools und Methoden zur Verfügung:

- DB2 Information - Unterstützung
  - Themen
  - Hilfe für DB2-Tools
  - Beispielprogramme
  - Lernprogramme
- Für den Download verfügbare PDF-Dateien, PDF-Dateien auf CD und gedruckte Bücher
  - Handbücher
  - Referenzhandbücher
- Befehlszeilenhilfe
  - Hilfe für Befehle
  - Hilfe für Nachrichten
  - Hilfe für SQL-Anweisungen
- Installierter Quellcode
  - Beispielprogramme

Darüber hinaus können Sie auf zusätzliche technische Informationen zu DB2 Universal Database™, wie beispielsweise technische Hinweise (Technotes), White Papers und Redbooks™, online über ibm.com® zugreifen. Rufen Sie die Website 'DB2 Information Management - Library' unter www.ibm.com/software/data/pubs/ auf.

Aktualisierungen der DB2-Dokumentation

In bestimmten Fällen stellt IBM® in regelmäßigen Abständen Dokumentations-Fix-Paks und andere Dokumentationsaktualisierungen für 'DB2 Information - Unterstützung' zur Verfügung. Wenn Sie über http://publib.boulder.ibm.com/infocenter/db2help/ auf 'DB2 Information - Unterstützung' zugreifen, erhalten Sie stets die neuesten Informationen. Falls Sie 'DB2 Information - Unterstützung' lokal installiert haben, müssen Sie alle Aktualisierungen manuell installieren, bevor Sie sie anzeigen können. Diese Dokumentationsaktualisierungen ermöglichen Ihnen, die Informationen, die Sie von der CD mit *DB2 Information - Unterstützung* installiert haben, auf den neuesten Stand zu bringen, sobald neue Informationen verfügbar sind.

'DB2 Information - Unterstützung' wird häufiger aktualisiert als die PDF- und Hardcopy-Bücher. Um stets die jeweils neuesten technischen Informationen zu DB2 zur Verfügung zu haben, sollten Sie die Dokumentationsaktualisierungen installieren, sobald sie verfügbar sind, oder 'DB2 Information - Unterstützung' über die Website www.ibm.com aufrufen.

Zugehörige Konzepte:

- „CLI sample programs" in *CLI Guide and Reference, Volume 1*
- „Java sample programs" in *Application Development Guide: Building and Running Applications*
- „DB2 Information - Unterstützung" auf Seite 686

Zugehörige Tasks:

- „Aufrufen der Kontexthilfe über ein DB2-Tool" auf Seite 705
- „Aktualisieren der auf Ihrem Computer oder Intranet-Server installierten Komponente 'DB2 Information - Unterstützung'" auf Seite 697
- „Aufrufen der Hilfe für Nachrichten über den Befehlszeilenprozessor" auf Seite 706
- „Aufrufen der Hilfe für Befehle über den Befehlszeilenprozessor" auf Seite 707
- „Aufrufen der Hilfe für den SQL-Status über den Befehlszeilenprozessor" auf Seite 707

Zugehörige Referenzen:

- „DB2-Dokumentation in PDF-Format und gedrucktem Format" auf Seite 698

DB2 Information - Unterstützung

Die DB2®-Komponente 'DB2 Information - Unterstützung' bietet Ihnen die Möglichkeit, auf alle Informationen zuzugreifen, die Sie zur optimalen Nutzung der Produkte innerhalb der DB2-Produktfamilie, wie z. B. DB2 Universal Database™, DB2 Connect™, DB2 Information Integrator und DB2 Query Patroller™, benötigen. 'DB2 Information - Unterstützung' dokumentiert auch die wichtigsten DB2-Funktionen und -Komponenten, einschließlich der Funktionen für die Replikation, das Data Warehousing und die DB2 Extender. Wenn Sie für die Anzeige von 'DB2 Information - Unterstützung' Mozilla ab Version 1.0 oder Microsoft® Internet Explorer ab Version 5.5 verwenden, stehen Ihnen die folgenden Funktionen zur Verfügung. Für bestimmte Funktionen muss die JavaScript™-Unterstützung aktiviert werden :

Flexible Installationsoptionen

Wählen Sie für die Anzeige der DB2-Dokumentation die Option, die Ihren Anforderungen am besten entspricht:

- Stellen Sie ohne großen Aufwand sicher, dass Ihre Dokumentation stets auf dem neuesten Stand ist, indem Sie auf die gesamte Dokumentation direkt über 'DB2 Information - Unterstützung' auf der IBM® Website unter http://publib.boulder.ibm.com/infocenter/db2help/ zugreifen.
- Reduzieren Sie den Aktualisierungsaufwand auf ein Minimum und begrenzen Sie den Datenaustausch auf Ihr Intranet, indem Sie die DB2-Dokumentation auf einem einzigen Server innerhalb Ihres Intranets installieren.
- Erzielen Sie maximale Flexibilität und reduzieren Sie die Abhängigkeit von Netzwerkverbindungen, indem Sie die DB2-Dokumentation auf dem eigenen Computer installieren.

Suchen

Sie können alle Themen in 'DB2 Information - Unterstützung' durchsuchen, indem Sie einen Suchbegriff im Textfeld **Suchen** eingeben. Schließen Sie Begriffe in Anführungszeichen ein, wenn Sie nach exakten Übereinstimmungen suchen möchten. Mit Hilfe von Platzhalterzeichen (*, ?) und Booleschen Operatoren (AND, NOT, OR) können Sie die Suche eingrenzen.

Aufgabenorientiertes Inhaltsverzeichnis

Die Themen in der DB2-Dokumentation können über ein zentrales Inhaltsverzeichnis lokalisiert werden. Das Inhaltsverzeichnis ist primär auf der Basis übergeordneter Aufgabenbereiche aufgebaut, enthält jedoch auch Einträge für Produktübersichten, Ziele, Referenzinformationen sowie einen Index und ein Glossar.

- Produktübersichten beschreiben die Beziehung zwischen den in der DB2-Produktfamilie verfügbaren Produkten sowie die von den einzelnen Produkten bereitgestellten Funktionen und enthalten darüber hinaus die neuesten Release-Informationen für diese Produkte.
- Aufgabenkategorien, wie z. B. Installation, Verwaltung und Entwicklung, umfassen Themen, mit deren Hilfe Sie die einzelnen Aufgaben schnell ausführen und sich außerdem genauere Kenntnisse über die Hintergrundinformationen zu diesen Aufgaben verschaffen können.
- In den Referenzthemen finden Sie detaillierte Informationen zu einem Thema, einschließlich der Anweisungs- und Befehlssyntax, der Hilfetexte zu Nachrichten und der Konfigurationsparameter.

Anzeigen des aktuellen Themas im Inhaltsverzeichnis

Wenn Sie sehen möchten, welchem Bereich des Inhaltsverzeichnisses das aktuelle Thema zugeordnet ist, klicken Sie den Knopf **Aktualisieren / aktuelles Thema anzeigen** im Teilfenster des Inhaltsverzeichnisses oder den Knopf **Im Inhaltsverzeichnis anzeigen** im Inhaltsteilfenster an. Diese Funktion ist zum Beispiel dann von Nutzen, wenn Sie mehreren Links zu zugehörigen Themen in verschiedenen Dateien gefolgt sind oder ein Thema über das Ergebnis einer Suche aufgerufen haben.

Index Über den Index können Sie auf die gesamte Dokumentation zugreifen. Der Index ist alphabetisch nach Indexeinträgen sortiert.

Glossar

Im Glossar finden Sie Definitionen zu Termini, die in der DB2-Dokumentation verwendet werden. Das Glossar ist alphabetisch nach Glossareinträgen sortiert.

Integrierte übersetzte Informationen

Die Informationen in 'DB2 Information - Unterstützung' werden in der Sprache angezeigt, die Sie in den Benutzervorgaben des verwendeten Browsers festgelegt haben. Ist ein Thema nicht in der bevorzugten Sprache verfügbar, wird die englische Version des Themas angezeigt.

Technische Informationen zu iSeries™ finden Sie im Informationszentrum von IBM eServer™ iSeries unter www.ibm.com/eserver/iseries/infocenter/.

Zugehörige Konzepte:

- „DB2 Information - Unterstützung: Installationsszenarios" auf Seite 688

Zugehörige Tasks:

- „Aktualisieren der auf Ihrem Computer oder Intranet-Server installierten Komponente 'DB2 Information - Unterstützung'" auf Seite 697
- „Anzeigen von Themen in der gewünschten Sprache in 'DB2 Information - Unterstützung'" auf Seite 698
- „Aufrufen von 'DB2 Information - Unterstützung'" auf Seite 695
- „Installation von 'DB2 Information - Unterstützung' mit dem DB2-Installationsassistenten (UNIX)" auf Seite 690
- „Installation von 'DB2 Information - Unterstützung' mit dem DB2-Installationsassistenten (Windows)" auf Seite 693

DB2 Information - Unterstützung: Installationsszenarios

Je nach Arbeitsumgebung kann es unterschiedliche Anforderungen hinsichtlich des Zugriffs auf DB2®-Informationen geben. Sie können auf 'DB2 Information - Unterstützung' entweder auf der IBM® Website zugreifen oder auf einem Server im unternehmensinternen Netzwerk oder auf eine auf dem lokalen Computer installierte Version. In allen drei Fällen befindet sich die Dokumentation in 'DB2 Information - Unterstützung', einem strukturierten System themenbasierter Informationen, die über einen Browser angezeigt werden können. Standardmäßig greifen DB2-Produkte auf 'DB2 Information - Unterstützung' auf der IBM Website zu. Wenn Sie jedoch auf 'DB2 Information - Unterstützung' auf einem Intranet-Server oder auf dem eigenen Computer zugreifen möchten, müssen Sie 'DB2 Information - Unterstützung' mit Hilfe der entsprechenden CD installieren, die sich im Programmpaket des Produkts befindet. Anhand der nachfolgenden Übersicht über die verfügbaren Optionen für den Zugriff auf die DB2-Dokumentation und mit Hilfe der drei Installationsszenarios können Sie ermitteln, welche Methode für den Zugriff auf 'DB2 Information - Unterstützung' für Ihre Anforderungen und Arbeitsumgebung am besten geeignet ist und welche Aspekte Sie bei der Installation berücksichtigen müssen.

Übersicht über die verfügbaren Optionen für den Zugriff auf die DB2-Dokumentation:

Die folgende Tabelle enthält Empfehlungen hinsichtlich der für Ihre Arbeitsumgebung geeigneten Optionen für den Zugriff auf die DB2-Produktdokumentation in 'DB2 Information - Unterstützung'.

Internetzugriff	Intranetzugriff	Empfehlung
Ja	Ja	Greifen Sie entweder über die IBM Website auf 'DB2 Information - Unterstützung' zu oder auf die auf einem Intranet-Server installierte Version von 'DB2 Information - Unterstützung'.
Ja	Nein	Greifen Sie über die IBM Website auf 'DB2 Information - Unterstützung' zu.
Nein	Ja	Greifen Sie auf die auf einem Intranet-Server installierte Version von 'DB2 Information - Unterstützung' zu.
Nein	Nein	Greifen Sie auf die auf einem lokalen Computer installierte Version von 'DB2 Information - Unterstützung' zu.

Szenario: Zugriff auf 'DB2 Information - Unterstützung' auf Ihrem Computer:

Tsu-Chen besitzt eine Fabrik in einer Kleinstadt, in der es vor Ort keinen Anbieter für einen Internetzugang gibt. Für die Verwaltung des Lagerbestands, der Produktbestellungen, der Betriebsausgaben und seines Bankkontos hat Tsu-Chen DB2 Universal Database™ gekauft. Da er zuvor noch nie ein DB2-Produkt verwendet hat, muss er anhand der DB2-Produktdokumentation lernen, wie die Verwaltung funktioniert.

Nachdem er DB2 Universal Database mit der Option für die Standardinstallation auf seinem Computer installiert hat, versucht Tsu-Chen, auf die DB2-Dokumentation zuzugreifen. Sein Browser zeigt jedoch eine Fehlernachricht mit der Information an, dass die Seite, die geöffnet werden sollte, nicht gefunden werden kann. Tsu-Chen überprüft das Installationshandbuch für sein DB2-Produkt und findet

heraus, dass er 'DB2 Information - Unterstützung' zunächst installieren muss, um auf seinem Computer auf die DB2-Dokumentation zugreifen zu können. Im Programmpaket findet er die *CD für DB2 Information - Unterstützung* und installiert sie.

Über das Programm zum Aufrufen von Anwendungen für sein Betriebssystem hat Tsu-Chen nun Zugriff auf 'DB2 Information - Unterstützung', um sich mit der Verwendung seines DB2-Produkts vertraut zu machen und so einen wertvollen Beitrag zum Erfolg seines Unternehmens leisten.

Szenario: Zugriff auf 'DB2 Information - Unterstützung' über die IBM Website:

Colin ist IT-Berater bei einer Schulungsfirma. Er ist auf Datenbanktechnologie und SQL spezialisiert und hält Seminare zu diesen Themen für Unternehmen aus ganz Nordamerika ab. Hierfür verwendet er DB2 Universal Database. Im Rahmen seiner Seminare verwendet Colin die DB2-Dokumentation als Unterrichtsmaterial. Für SQL-Kurse beispielsweise verwendet Colin die DB2-Dokumentation zu SQL, um die grundlegende und erweiterte Syntax für Datenbankabfragen zu unterrichten.

Die meisten Unternehmen, bei denen Colin unterrichtet, verfügen über einen Internetzugang. Aus diesem Grund entschied sich Colin, seinen tragbaren Computer für den Zugriff auf 'DB2 Information - Unterstützung' über die Website von IBM zu konfigurieren, als er die letzte Version von DB2 Universal Database installiert hat. Diese Konfiguration ermöglicht es Colin, während seiner Seminare online auf die neueste DB2-Dokumentation zuzugreifen.

Wenn er auf Reisen ist, hat Colin bisweilen allerdings keinen Internetzugang. Dieser Umstand war für ihn recht problematisch, insbesondere dann, wenn er Zugriff auf die DB2-Dokumentation benötigte, um sich auf seine Seminare vorzubereiten. Um Situationen wie diese zu vermeiden, installierte Colin eine Kopie von 'DB2 Information - Unterstützung' auf seinem tragbaren Computer.

Auf diese Weise hat Colin nun jederzeit eine Kopie der DB2-Dokumentation zur Verfügung und ist dadurch wesentlich flexibler. Mit dem Befehl **db2set** kann Colin ohne Schwierigkeiten die Registrierdatenbankvariablen auf seinem tragbaren Computer so konfigurieren, dass er den jeweiligen Umständen entsprechend entweder über die Website von IBM oder über seinen tragbaren Computer auf 'DB2 Information - Unterstützung' zugreifen kann.

Szenario: Zugriff auf 'DB2 Information - Unterstützung' über einen Intranet-Server:

Eva arbeitet als leitende Datenbankadministratorin für eine Lebensversicherung. In ihre Zuständigkeit fallen auch das Installieren und Konfigurieren der neuesten Version von DB2 Universal Database auf den UNIX®-basierten Datenbankservern des Unternehmens. Vor Kurzem hat das Unternehmen seine Mitarbeiter darüber informiert, dass sie aus Sicherheitsgründen während der Arbeitszeit keinen Internetzugriff erhalten würden. Da ihr Unternehmen in einer Netzwerkumgebung arbeitet, beschließt Eva, eine Kopie von 'DB2 Information - Unterstützung' auf einem Intranet-Server zu installieren, damit alle Mitarbeiter, die das Data Warehouse des Unternehmens regelmäßig verwenden (Vertriebsbeauftragte, Vertriebsleiter und Geschäftsanalysten), Zugriff auf die DB2-Dokumentation haben.

Eva weist ihr Datenbankteam an, die neueste Version von DB2 Universal Database auf allen Computern der Mitarbeiter mit Hilfe einer Antwortdatei zu installieren, um sicherzustellen, dass die Konfiguration des Zugriffs auf 'DB2 Information - Unterstützung' auf allen Computern mit dem Hostnamen und der Portnummer des Intranet-Servers erfolgt.

Durch ein Missverständnis installiert jedoch Migual, ein Datenbankadministrator in Evas Team, eine Kopie von 'DB2 Information - Unterstützung' auf mehreren Mitarbeitercomputern, anstatt DB2 Universal Database für den Zugriff auf 'DB2 Information - Unterstützung' über den Intranet-Server zu konfigurieren. Um diesen Fehler zu korrigieren, weist Eva Migual an, mit dem Befehl **db2set** die Registrierdatenbankvariablen von 'DB2 Information - Unterstützung' (DB2_DOCHOST für den Hostnamen und DB2_DOCPORT für die Portnummer) auf allen entsprechenden Computern zu ändern. Anschließend haben nun alle erforderlichen Computer im Netzwerk Zugriff auf 'DB2 Information - Unterstützung', und die Mitarbeiter können mit Hilfe der DB2-Dokumentation Antworten auf ihre Fragen zu DB2 finden.

Zugehörige Konzepte:

- „DB2 Information - Unterstützung" auf Seite 686

Zugehörige Tasks:

- „Aktualisieren der auf Ihrem Computer oder Intranet-Server installierten Komponente 'DB2 Information - Unterstützung'" auf Seite 697
- „Installation von 'DB2 Information - Unterstützung' mit dem DB2-Installationsassistenten (UNIX)" auf Seite 690
- „Installation von 'DB2 Information - Unterstützung' mit dem DB2-Installationsassistenten (Windows)" auf Seite 693
- „Festlegen der Speicherposition für den Zugriff auf 'DB2 Information - Unterstützung': Gemeinsame GUI - Hilfe"

Zugehörige Referenzen:

- „db2set - DB2 Profile Registry Command" in *Command Reference*

Installation von 'DB2 Information - Unterstützung' mit dem DB2-Installationsassistenten (UNIX)

Es gibt drei Möglichkeiten, auf die DB2-Produktdokumentation zuzugreifen: auf der IBM Website, auf einem Intranet-Server oder auf eine auf dem lokalen Computer installierte Version. Standardmäßig greifen DB2-Produkte auf die DB2-Dokumentation auf der IBM Website zu. Wenn Sie jedoch auf die DB2-Dokumentation auf einem Intranet-Server oder auf dem eigenen Computer zugreifen möchten, müssen Sie die Dokumentation von der *CD 'DB2 Information - Unterstützung'* aus installieren. Mit dem DB2-Installationsassistenten können Sie Ihre Installationseinstellungen definieren und 'DB2 Information - Unterstützung' auf einem Computer installieren, der das Betriebssystem UNIX verwendet.

Voraussetzungen:

Dieser Abschnitt erläutert die Voraussetzungen für Hardware, Betriebssystem, Software und Kommunikation zum Installieren von 'DB2 Information - Unterstützung' auf UNIX-Computern.

- **Hardwarevoraussetzungen**

  Sie benötigen einen der folgenden Prozessoren:
  - PowerPC (AIX)
  - HP 9000 (HP-UX)
  - Intel 32-Bit (Linux)
  - Solaris UltraSPARC-Computer (Solaris-Betriebsumgebung)
- **Betriebssystemvoraussetzungen**

  Sie benötigen eines der folgenden Betriebssysteme:
  - IBM AIX 5.1 (auf PowerPC)
  - HP-UX 11i (auf HP 9000)
  - Red Hat Linux 8.0 (auf Intel 32–Bit)
  - SuSE Linux 8.1 (auf Intel 32–Bit)
  - Sun Solaris Version 8 (auf UltraSPARC-Computern in der Solaris-Betriebsumgebung)

  Anmerkung: 'DB2 Information - Unterstützung' kann unter einem Teil der UNIX-Betriebssysteme ausgeführt werden, unter denen DB2-Clients unterstützt werden. Daher wird empfohlen, entweder über die IBM Website auf 'DB2 Information - Unterstützung' zuzugreifen oder 'DB2 Information - Unterstützung' auf einem Intranet-Server zu installieren und dort auf die Dokumentation zuzugreifen.
- **Softwarevoraussetzungen**
  - Unterstützte Browser:
    - Mozilla Version 1.0 oder höher
- Beim DB2-Installationsassistenten handelt es sich um ein grafisches Installationsprogramm. Um den DB2-Installationsassistenten auf Ihrem Computer ausführen zu können, benötigen Sie eine Implementierung der X Window System-Software zur Wiedergabe einer grafischen Benutzerschnittstelle (GUI). Bevor Sie den DB2-Installationsassistenten ausführen können, müssen Sie die entsprechende Anzeigefunktion (DISPLAY) unbedingt ordnungsgemäß exportieren. Geben Sie hierzu beispielsweise den folgenden Befehl an der Eingabeaufforderung ein:

  ```
  export DISPLAY=9.26.163.144:0.
  ```
- **Kommunikationsvoraussetzungen**
  - TCP/IP

Vorgehensweise:

Um 'DB2 Information - Unterstützung' mit Hilfe des DB2-Installationsassistenten zu installieren, gehen Sie wie folgt vor:

1. Melden Sie sich am System an.
2. Legen Sie die Produkt-CD von 'DB2 Information - Unterstützung' in das CD-Laufwerk ein, und hängen Sie die CD an Ihr System an.
3. Wechseln Sie in das Verzeichnis, in dem die CD angehängt ist. Geben Sie hierzu den folgenden Befehl ein:

   ```
   cd /cd
   ```

   Hierbei steht */cd* für den Mountpunkt der CD.
4. Geben Sie den Befehl **./db2setup** ein, um den DB2-Installationsassistenten zu starten.
5. Die IBM DB2-Klickstartleiste wird geöffnet. Um direkt mit der Installation von 'DB2 Information - Unterstützung' fortzufahren, klicken Sie **Produkt installieren** an. Die Onlinehilfe enthält Informationen, die Sie durch die verbleibenden

Schritte der Installation führen. Um die Onlinehilfe aufzurufen, klicken Sie **Hilfe** an. Sie können jederzeit **Abbrechen** anklicken, um die Installation zu beenden.

6. Klicken Sie im Fenster **Wählen Sie das zu installierende Produkt aus** den Knopf **Weiter** an.
7. Klicken Sie **Weiter** im Fenster **Willkommen beim DB2-Installationsassistenten** an. Der DB2-Installationsassistent leitet Sie durch die erforderlichen Schritte zum Installieren des Programms.
8. Um mit der Installation fortfahren zu können, müssen Sie die Lizenzvereinbarung akzeptieren. Wählen Sie auf der Seite **Lizenzvereinbarung** die Option **Bedingungen in der Lizenzvereinbarung anerkennen** aus, und klicken Sie **Weiter** an.
9. Wählen Sie **DB2 Information - Unterstützung auf diesem Computer installieren** auf der Seite **Installationsaktion auswählen** aus. Wenn Sie 'DB2 Information - Unterstützung' zu einem späteren Zeitpunkt auf diesem Computer oder anderen Computern mit Hilfe einer Antwortdatei installieren möchten, wählen Sie **Ihre Einstellungen in einer Antwortdatei speichern** aus. Klicken Sie **Weiter** an.
10. Wählen Sie auf der Seite **Zu installierende Sprachen auswählen** die Sprachen aus, in denen 'DB2 Information - Unterstützung' installiert werden soll. Klicken Sie den Knopf **Weiter** an.
11. Konfigurieren Sie 'DB2 Information - Unterstützung' auf der Seite **Port von DB2 Information - Unterstützung angeben** für eingehende Kommunikation. Klicken Sie **Weiter** an, um mit der Installation fortzufahren.
12. Überprüfen Sie auf der Seite **Kopieren der Dateien starten** noch einmal die von Ihnen ausgewählten Installationseinstellungen. Wenn Sie die Einstellungen ändern möchten, klicken Sie **Zurück** an. Klicken Sie **Installieren** an, um die Dateien von 'DB2 Information - Unterstützung' auf Ihren Computer zu kopieren.

Sie können 'DB2 Information - Unterstützung' auch mit Hilfe einer Antwortdatei installieren.

Die Installationsprotokolldateien `db2setup.his`, `db2setup.log` und `db2setup.err` befinden sich standardmäßig im Verzeichnis `/tmp`.

Die Datei `db2setup.log` erfasst alle Installationsinformationen zu DB2-Produkten, einschließlich Fehlern. Die Datei `db2setup.his` zeichnet alle DB2-Produktinstallationen auf Ihrem Computer auf. DB2 hängt die Datei `db2setup.log` an die Datei `db2setup.his` an. Die Datei `db2setup.err` erfasst die gesamte Fehlerausgabe, die von Java zurückgegeben wird, wie beispielsweise Informationen zu Ausnahmebedingungen und Traps.

Nach Abschluss der Installation ist 'DB2 Information - Unterstützung' je nach UNIX-Betriebssystem in einem der folgenden Verzeichnisse installiert:

- AIX: /usr/opt/db2_08_01
- HP-UX: /opt/IBM/db2/V8.1
- Linux: /opt/IBM/db2/V8.1
- Solaris-Betriebsumgebung: /opt/IBM/db2/V8.1

Zugehörige Konzepte:

- „DB2 Information - Unterstützung" auf Seite 686
- „DB2 Information - Unterstützung: Installationsszenarios" auf Seite 688

Zugehörige Tasks:

- „Installieren von DB2 mit Hilfe einer Antwortdatei (UNIX)" in *Installation und Konfiguration Ergänzung*
- „Aktualisieren der auf Ihrem Computer oder Intranet-Server installierten Komponente 'DB2 Information - Unterstützung'" auf Seite 697
- „Anzeigen von Themen in der gewünschten Sprache in 'DB2 Information - Unterstützung'" auf Seite 698
- „Aufrufen von 'DB2 Information - Unterstützung'" auf Seite 695
- „Installation von 'DB2 Information - Unterstützung' mit dem DB2-Installationsassistenten (Windows)" auf Seite 693

Installation von 'DB2 Information - Unterstützung' mit dem DB2-Installationsassistenten (Windows)

Es gibt drei Möglichkeiten, auf die DB2-Produktdokumentation zuzugreifen: auf der IBM Website, auf einem Intranet-Server oder auf eine auf dem lokalen Computer installierte Version. Standardmäßig greifen DB2-Produkte auf die DB2-Dokumentation auf der IBM Website zu. Wenn Sie jedoch auf die DB2-Dokumentation auf einem Intranet-Server oder auf dem eigenen Computer zugreifen möchten, müssen Sie die DB2-Dokumentation von der *CD 'DB2 Information - Unterstützung'* aus installieren. Mit dem DB2-Installationsassistenten können Sie Ihre Installationseinstellungen definieren und 'DB2 Information - Unterstützung' auf einem Computer installieren, der ein Windows-Betriebssystem verwendet.

Voraussetzungen:

Dieser Abschnitt erläutert die Voraussetzungen für Hardware, Betriebssystem, Software und Kommunikation zum Installieren von 'DB2 Information - Unterstützung' unter Windows.

- **Hardwarevoraussetzungen**

  Sie benötigen einen der folgenden Prozessoren:

  – 32-Bit-Computer: eine Pentium- oder mit Pentium kompatible CPU
- **Betriebssystemvoraussetzungen**

  Sie benötigen eines der folgenden Betriebssysteme:

  – Windows 2000

  – Windows XP

  Anmerkung: 'DB2 Information - Unterstützung' kann unter einem Teil der Windows-Betriebssysteme ausgeführt werden, unter denen DB2-Clients unterstützt werden. Daher wird empfohlen, entweder über die IBM Website auf 'DB2 Information - Unterstützung' zuzugreifen oder 'DB2 Information - Unterstützung' auf einem Intranet-Server zu installieren und dort auf die Dokumentation zuzugreifen.
- **Softwarevoraussetzungen**

  – Unterstützte Browser:

    - Mozilla 1.0 oder höher

    - Internet Explorer Version 5.5 oder 6.0 (Version 6.0 für Windows XP)
- **Kommunikationsvoraussetzungen**

  – TCP/IP

Einschränkungen:

- Sie benötigen einen Benutzereintrag mit Administratorberechtigung, um 'DB2 Information - Unterstützung' zu installieren.

Vorgehensweise:

Um 'DB2 Information - Unterstützung' mit Hilfe des DB2-Installationsassistenten zu installieren, gehen Sie wie folgt vor:

1. Melden Sie sich mit dem für die Installation von 'DB2 Information - Unterstützung' definierten Benutzereintrag am System an.
2. Legen Sie die CD in das Laufwerk ein. Die IBM DB2 Setup-Klickstartleiste wird von der Funktion für automatische Ausführung gestartet, sofern diese Funktion aktiviert ist.
3. Der DB2-Installationsassistent ermittelt die Systemsprache und startet das Installationsprogramm für diese Sprache. Wenn Sie das Installationsprogramm nicht in Englisch ausführen möchten oder wenn beim automatischen Starten des Programms ein Fehler aufgetreten ist, können Sie den DB2-Installationsassistenten auch manuell starten.

   Um den DB2-Installationsassistenten manuell zu starten, gehen Sie wie folgt vor:

   a. Klicken Sie **Start** an, und wählen Sie die Option **Ausführen** aus.
   b. Geben Sie im Feld **Öffnen** den folgenden Befehl ein:

      ```
      x:\setup.exe /i zweistellige sprachenkennung
      ```

      Hierbei steht *x:* für das CD-Laufwerk und *zweistellige sprachenkennung* für die Sprache, in der das Installationsprogramm ausgeführt werden soll.
   c. Klicken Sie **OK** an.
4. Die IBM DB2-Klickstartleiste wird geöffnet. Um direkt mit der Installation von 'DB2 Information - Unterstützung' fortzufahren, klicken Sie **Produkt installieren** an. Die Onlinehilfe enthält Informationen, die Sie durch die verbleibenden Schritte der Installation führen. Um die Onlinehilfe aufzurufen, klicken Sie **Hilfe** an. Sie können jederzeit **Abbrechen** anklicken, um die Installation zu beenden.
5. Klicken Sie im Fenster **Wählen Sie das zu installierende Produkt aus** den Knopf **Weiter** an.
6. Klicken Sie **Weiter** im Fenster **Willkommen beim DB2-Installationsassistenten** an. Der DB2-Installationsassistent leitet Sie durch die erforderlichen Schritte zum Installieren des Programms.
7. Um mit der Installation fortfahren zu können, müssen Sie die Lizenzvereinbarung akzeptieren. Wählen Sie auf der Seite **Lizenzvereinbarung** die Option **Bedingungen in der Lizenzvereinbarung anerkennen** aus, und klicken Sie **Weiter** an.
8. Wählen Sie **DB2 Information - Unterstützung auf diesem Computer installieren** auf der Seite **Installationsaktion auswählen** aus. Wenn Sie 'DB2 Information - Unterstützung' zu einem späteren Zeitpunkt auf diesem Computer oder anderen Computern mit Hilfe einer Antwortdatei installieren möchten, wählen Sie **Ihre Einstellungen in einer Antwortdatei speichern** aus. Klicken Sie **Weiter** an.
9. Wählen Sie auf der Seite **Zu installierende Sprachen auswählen** die Sprachen aus, in denen 'DB2 Information - Unterstützung' installiert werden soll. Klicken Sie den Knopf **Weiter** an.

10. Konfigurieren Sie 'DB2 Information - Unterstützung' auf der Seite **Port von DB2 Information - Unterstützung angeben** für eingehende Kommunikation. Klicken Sie **Weiter** an, um mit der Installation fortzufahren.
11. Überprüfen Sie auf der Seite **Kopieren der Dateien starten** noch einmal die von Ihnen ausgewählten Installationseinstellungen. Wenn Sie die Einstellungen ändern möchten, klicken Sie **Zurück** an. Klicken Sie **Installieren** an, um die Dateien von 'DB2 Information - Unterstützung' auf Ihren Computer zu kopieren.

Sie haben die Möglichkeit, 'DB2 Information - Unterstützung' mit Hilfe einer Antwortdatei zu installieren. Sie können auch den Befehl **db2rspgn** verwenden, um eine Antwortdatei auf der Grundlage einer vorhandenen Installation zu generieren.

Die Dateien `db2.log` und `db2wi.log` im Verzeichnis `'Eigene Dateien'\DB2LOG\` enthalten Informationen zu Fehlern, die während der Installation aufgetreten sind. Die Position des Verzeichnisses `'Eigene Dateien'` hängt von den Einstellungen Ihres Computers ab.

Die Datei `db2wi.log` erfasst die neuesten DB2-Installationsinformationen. Die Datei `db2.log` erfasst die Protokollinformationen von DB2-Produktinstallationen.

Zugehörige Konzepte:

- „DB2 Information - Unterstützung" auf Seite 686
- „DB2 Information - Unterstützung: Installationsszenarios" auf Seite 688

Zugehörige Tasks:

- „Installieren eines DB2-Produkts mit Hilfe einer Antwortdatei (Windows)" in *Installation und Konfiguration Ergänzung*
- „Aktualisieren der auf Ihrem Computer oder Intranet-Server installierten Komponente 'DB2 Information - Unterstützung'" auf Seite 697
- „Anzeigen von Themen in der gewünschten Sprache in 'DB2 Information - Unterstützung'" auf Seite 698
- „Aufrufen von 'DB2 Information - Unterstützung'" auf Seite 695
- „Installation von 'DB2 Information - Unterstützung' mit dem DB2-Installationsassistenten (UNIX)" auf Seite 690

Zugehörige Referenzen:

- „db2rspgn - Response File Generator Command (Windows)" in *Command Reference*

Aufrufen von 'DB2 Information - Unterstützung'

'DB2 Information - Unterstützung' bietet Ihnen die Möglichkeit, auf alle Informationen zuzugreifen, die Sie zur Verwendung der DB2-Produkte für die Betriebssysteme Linux, UNIX und Windows, wie z. B. DB2 Universal Database, DB2 Connect, DB2 Information Integrator und DB2 Query Patroller, benötigen.

Rufen Sie 'DB2 Information - Unterstützung' auf eine der folgenden Arten auf:

- Von einem Computer aus, auf dem ein DB2 UDB-Client oder -Server installiert ist
- Von einem Intranet-Server oder einem lokalen Computer aus, auf dem 'DB2 Information - Unterstützung' installiert ist

- Über die IBM Website

Voraussetzungen:

Führen Sie vor dem Aufrufen von 'DB2 Information - Unterstützung' folgende Schritte aus:

- *Optional*: Konfigurieren des Browsers für die Anzeige der Themen in der gewünschten Landessprache
- *Optional*: Konfigurieren des DB2-Clients für die Verwendung der auf Ihrem Computer oder Intranet-Server installierten Komponente 'DB2 Information - Unterstützung'

Vorgehensweise:

Gehen Sie wie folgt vor, um 'DB2 Information - Unterstützung' auf einem Computer aufzurufen, auf dem ein DB2 UDB-Client oder -Server installiert ist:

- Wählen Sie (unter Windows) **Start ⟶ Programme ⟶ IBM DB2 ⟶ Information ⟶ DB2 Information - Unterstützung** aus.
- Geben Sie in der Befehlszeile Folgendes ein:
  - Unter Linux und UNIX: Geben Sie den Befehl **db2icdocs** ein.
  - Unter Windows: Geben Sie den Befehl **db2icdocs.exe** ein.

Gehen Sie wie folgt vor, um die auf einem Intranet-Server oder lokalen Computer installierte Komponente 'DB2 Information - Unterstützung' in einem Webbrowser zu öffnen:

- Öffnen Sie die Webseite unter http://<hostname>:<portnummer>/. Dabei stellt <hostname> den Namen des Hosts dar und <portnummer> die Nummer des Ports, an dem 'DB2 Information - Unterstützung' verfügbar ist.

Gehen Sie wie folgt vor, um 'DB2 Information - Unterstützung' auf der IBM Website in einem Webbrowser zu öffnen:

- Öffnen Sie die Webseite unter publib.boulder.ibm.com/infocenter/db2help/.

Zugehörige Konzepte:

- „DB2 Information - Unterstützung" auf Seite 686
- „DB2 Information - Unterstützung: Installationsszenarios" auf Seite 688

Zugehörige Tasks:

- „Anzeigen von Themen in der gewünschten Sprache in 'DB2 Information - Unterstützung'" auf Seite 698
- „Aufrufen der Kontexthilfe über ein DB2-Tool" auf Seite 705
- „Aktualisieren der auf Ihrem Computer oder Intranet-Server installierten Komponente 'DB2 Information - Unterstützung'" auf Seite 697
- „Aufrufen der Hilfe für Befehle über den Befehlszeilenprozessor" auf Seite 707
- „Festlegen der Speicherposition für den Zugriff auf 'DB2 Information - Unterstützung': Gemeinsame GUI - Hilfe"

Zugehörige Referenzen:

- „HELP Command" in *Command Reference*

Aktualisieren der auf Ihrem Computer oder Intranet-Server installierten Komponente 'DB2 Information - Unterstützung'

Die Komponente 'DB2 Information - Unterstützung', auf die Sie über http://publib.boulder.ibm.com/infocenter/db2help/ zugreifen können, wird in regelmäßigen Abständen durch neue oder geänderte Dokumentationen aktualisiert. IBM stellt in bestimmten Fällen auch Aktualisierungen von 'DB2 Information - Unterstützung' zum Download bereit, die Sie auf Ihrem Computer oder Intranet-Server installieren können. Durch die Aktualisierung von 'DB2 Information - Unterstützung' werden keine DB2-Client- oder -Serverprodukte aktualisiert.

Voraussetzungen:

Sie benötigen Zugriff auf einen Computer, der über eine Verbindung zum Internet verfügt.

Vorgehensweise:

Gehen Sie wie folgt vor, um die auf Ihrem Computer bzw. Intranet-Server installierte Komponente 'DB2 Information - Unterstützung' zu aktualisieren:

1. Öffnen Sie 'DB2 Information - Unterstützung' auf der IBM Website unter http://publib.boulder.ibm.com/infocenter/db2help/.
2. Klicken Sie im Downloadbereich der Eingangsseite den Link **DB2 Universal Database-Dokumentation** unter der Überschrift für Service und Unterstützung an.
3. Stellen Sie fest, ob die Version der installierten Komponente 'DB2 Information - Unterstützung' veraltet ist, indem Sie die Stufe des neuesten aktualisierten Dokumentationsimage mit der installierten Dokumentationsstufe vergleichen. Die installierte Dokumentationsstufe ist auf der Eingangsseite von 'DB2 Information - Unterstützung' aufgeführt.
4. Wenn eine neuere Version von 'DB2 Information - Unterstützung' verfügbar ist, laden Sie das neueste aktualisierte Image für *DB2 Information - Unterstützung* für das von Ihnen verwendete Betriebssystem herunter.
5. Befolgen Sie zur Installation des aktualisierten Image für *DB2 Information - Unterstützung* die Anweisungen auf der Webseite.

Zugehörige Konzepte:

- „DB2 Information - Unterstützung: Installationsszenarios" auf Seite 688

Zugehörige Tasks:

- „Aufrufen von 'DB2 Information - Unterstützung'" auf Seite 695
- „Installation von 'DB2 Information - Unterstützung' mit dem DB2-Installationsassistenten (UNIX)" auf Seite 690
- „Installation von 'DB2 Information - Unterstützung' mit dem DB2-Installationsassistenten (Windows)" auf Seite 693

Anzeigen von Themen in der gewünschten Sprache in 'DB2 Information - Unterstützung'

In 'DB2 Information - Unterstützung werden Themen, wenn möglich, in der Sprache angezeigt, die in den Vorgaben Ihres Browsers angegeben ist. Falls ein Thema nicht in die gewünschte Sprache übersetzt wurde, wird es in 'DB2 Information - Unterstützung' in Englisch angezeigt.

Vorgehensweise:

Um Themen in der gewünschten Sprache im Browser 'Internet Explorer' anzuzeigen, gehen Sie wie folgt vor:

1. Klicken Sie im Internet Explorer **Extras** —> **Internetoptionen...** —> **Sprachen...** an. Das Fenster **Spracheinstellung** wird geöffnet.
2. Stellen Sie sicher, dass die gewünschte Sprache als erster Eintrag in der Liste angegeben ist.
   - Klicken Sie den Knopf **Hinzufügen...** an, um eine neue Sprache zur Liste hinzuzufügen.

     Anmerkung: Das Hinzufügen einer Sprache bedeutet nicht zwangsläufig, dass der Computer über die erforderlichen Schriftarten verfügt, um die Themen in der gewünschten Sprache anzuzeigen.
   - Um eine Sprache an den Anfang der Liste zu verschieben, wählen Sie zunächst die gewünschte Sprache und anschließend den Knopf **Nach oben** aus, bis die Sprache an erster Stelle in der Liste steht.
3. Aktualisieren Sie die Seite, um 'DB2 Information - Unterstützung' in der gewünschten Sprache anzuzeigen.

Um Themen in der gewünschten Sprache im Browser 'Mozilla' anzuzeigen, gehen Sie wie folgt vor:

1. Wählen Sie in Mozilla **Bearbeiten** —> **Einstellungen** —> **Sprachen** aus. Die Anzeige für die Auswahl der Sprache wird im Fenster mit den Einstellungen aufgerufen.
2. Stellen Sie sicher, dass die gewünschte Sprache als erster Eintrag in der Liste angegeben ist.
   - Wenn Sie eine neue Sprache hinzufügen möchten, klicken Sie den Knopf **Hinzufügen...** an, um eine Sprache im entsprechenden Fenster auszuwählen.
   - Um eine Sprache an den Anfang der Liste zu verschieben, wählen Sie zunächst die gewünschte Sprache und anschließend den Knopf **Nach oben** aus, bis die Sprache an erster Stelle in der Liste steht.
3. Aktualisieren Sie die Seite, um 'DB2 Information - Unterstützung' in der gewünschten Sprache anzuzeigen.

Zugehörige Konzepte:

- „DB2 Information - Unterstützung" auf Seite 686

DB2-Dokumentation in PDF-Format und gedrucktem Format

In den folgenden Tabellen sind die offiziellen Buchtitel, Formularnummern und PDF-Dateinamen aufgeführt. Zum Bestellen von Hardcopybüchern benötigen Sie den offiziellen Buchtitel. Zum Drucken der PDF-Version benötigen Sie den PDF-Dateinamen.

Die DB2-Dokumentation ist in die folgenden Kategorien unterteilt:

- DB2-Kerninformationen
- Verwaltungsinformationen
- Informationen zur Anwendungsentwicklung
- Informationsmanagement
- Informationen zu DB2 Connect
- Einführungsinformationen
- Lernprogramminformationen
- Informationen zu Zusatzkomponenten
- Release-Informationen

In den folgenden Tabellen wird für die einzelnen Bücher der DB2-Bibliothek beschrieben, welche Informationen zum Bestellen von Hardcopys bzw. zum Drucken oder Anzeigen der PDF-Versionen erforderlich sind. Eine vollständige Beschreibung der in der DB2-Bibliothek verfügbaren Bücher finden Sie im IBM Publications Center unter folgender Adresse: www.ibm.com/shop/publications/order.

DB2-Kerninformationen

Diese Bücher enthalten grundlegende Informationen für alle DB2-Benutzer. Diese Informationen sind sowohl für Programmierer als auch für Datenbankadministratoren geeignet und unterstützen Sie bei der Arbeit mit DB2 Connect, DB2 Warehouse Manager und anderen DB2-Produkten.

Tabelle 73. DB2-Kerninformationen

Name	IBM Form	PDF-Dateiname
IBM DB2 Universal Database Command Reference	SC09-4828	db2n0e81
IBM DB2 Universal Database Glossar	Keine Formnummer	db2t0g81
IBM DB2 Universal Database Fehlernachrichten, Band 1	GC12-3043, nicht als Hardcopy verfügbar	db2m1g81
IBM DB2 Universal Database Fehlernachrichten, Band 2	GC12-3042, nicht als Hardcopy verfügbar	db2m2g81
IBM DB2 Universal Database Neue Funktionen	SC12-3044	db2q0g81

Verwaltungsinformationen

Die Informationen in diesen Büchern umfassen die Themen, die zum effektiven Entwerfen, Implementieren und Verwalten von DB2-Datenbanken, Data Warehouses und Systemen zusammengeschlossener Datenbanken erforderlich sind.

Tabelle 74. Verwaltungsinformationen

Name	IBM Form	PDF-Dateiname
IBM DB2 Universal Database Systemverwaltung: Konzept	SC12-3057	db2d1g81
IBM DB2 Universal Database Systemverwaltung: Implementierung	SC12-3059	db2d2g81

Tabelle 74. Verwaltungsinformationen (Forts.)

Name	IBM Form	PDF-Dateiname
IBM DB2 Universal Database Systemverwaltung: Optimierung	SC12-3058	db2d3g81
IBM DB2 Universal Database Administrative API Reference	SC09-4824	db2b0e81
IBM DB2 Universal Database Dienstprogramme für das Versetzen von Daten Handbuch und Referenz	SC12-3055	db2dmg81
IBM DB2 Universal Database Datenwiederherstellung und hohe Verfügbarkeit Handbuch und Referenz	SC12-3054	db2hag81
IBM DB2 Universal Database Data Warehouse-Zentrale Verwaltung	SC12-3068	db2ddg81
IBM DB2 Universal Database SQL Reference, Volume 1	SC09-4844	db2s1e81
IBM DB2 Universal Database SQL Reference, Volume 2	SC09-4845	db2s2e81
IBM DB2 Universal Database System Monitor Guide and Reference	SC09-4847	db2f0e81

Informationen zur Anwendungsentwicklung

Die Informationen in diesen Büchern sind besonders für Anwendungsentwickler und Programmierer von Interesse, die mit DB2 Universal Database (DB2 UDB) arbeiten. Sie finden hier Informationen zu den unterstützten Programmiersprachen und Compilern sowie die Dokumentation, die für den Zugriff auf DB2 UDB über die verschiedenen unterstützten Programmierschnittstellen, z. B. eingebettetes SQL, ODBC, JDBC, SQLJ und CLI, erforderlich ist. Wenn Sie die Komponente 'DB2 Information - Unterstützung' verwenden, können Sie auch auf HTML-Versionen des Quellcodes für die Beispielprogramme zugreifen.

Tabelle 75. Informationen zur Anwendungsentwicklung

Name	IBM Form	PDF-Dateiname
IBM DB2 Universal Database Application Development Guide: Building and Running Applications	SC09-4825	db2axe81
IBM DB2 Universal Database Application Development Guide: Programming Client Applications	SC09-4826	db2a1e81
IBM DB2 Universal Database Application Development Guide: Programming Server Applications	SC09-4827	db2a2e81
IBM DB2 Universal Database Call Level Interface Guide and Reference, Volume 1	SC09-4849	db2l1e81

Tabelle 75. Informationen zur Anwendungsentwicklung (Forts.)

Name	IBM Form	PDF-Dateiname
IBM DB2 Universal Database Call Level Interface Guide and Reference, Volume 2	SC09-4850	db2l2e81
IBM DB2 Universal Database Data Warehouse Center Application Integration Guide	SC27-1124	db2ade81
IBM DB2 XML Extender Verwaltung und Programmierung	SC12-3062	db2sxg81

Informationsmanagement

Die Informationen in diesen Büchern beschreiben den Einsatz von Komponenten, mit denen Sie die Data Warehousing- und Analysefunktionen von DB2 Universal Database erweitern können.

Tabelle 76. Informationsmanagement

Name	IBM Form	PDF-Dateiname
IBM DB2 Warehouse Manager Standard Edition Informationskatalogzentrale Verwaltung	SC12-3070	db2dig81
IBM DB2 Warehouse Manager Standard Edition Installation	GC12-3069	db2idg81
IBM DB2 Warehouse Manager Standard Edition Managing ETI Solution Conversion Programs with DB2 Warehouse Manager	SC18-7727	iwhe1mste80

Informationen zu DB2 Connect

Die Informationen in dieser Kategorie beschreiben den Zugriff auf Daten auf großen und mittleren Serversystemen mit Hilfe von DB2 Connect Enterprise Edition oder DB2 Connect Personal Edition.

Tabelle 77. Informationen zu DB2 Connect

Name	IBM Form	PDF-Dateiname
IBM Konnektivität Ergänzung	Keine Formnummer	db2h1g81
IBM DB2 Connect Enterprise Edition Einstieg	GC12-3051	db2c6g81
IBM DB2 Connect Personal Edition Einstieg	GC12-3049	db2c1g81
IBM DB2 Connect Benutzerhandbuch	SC12-3048	db2c0g81

Einführungsinformationen

Die Informationen in dieser Kategorie unterstützen Sie beim Installieren und Konfigurieren von Servern, Clients und anderen DB2-Produkten.

Tabelle 78. Einführungsinformationen

Name	IBM Form	PDF-Dateiname
IBM DB2 Universal Database für DB2-Clients Einstieg	GC12-3052, nicht als Hardcopy verfügbar	db2itg81
IBM DB2 Universal Database für DB2-Server Einstieg	GC12-3047	db2isg81
IBM DB2 Universal Database Personal Edition Einstieg	GC12-3045	db2i1g81
IBM DB2 Universal Database Installation und Konfiguration Ergänzung	GC12-3046, nicht als Hardcopy verfügbar	db2iyg81
IBM DB2 Universal Database Data Links Manager Einstieg	GC12-3056	db2z6g81

Lernprogramminformationen

In den Lernprogramminformationen werden DB2-Funktionen vorgestellt. Darüber hinaus wird die Ausführung verschiedener Tasks beschrieben.

Tabelle 79. Lernprogramminformationen

Name	IBM Form	PDF-Dateiname
Lernprogramm für das Informationsmanagement: Data Warehouse - Einführung	Keine Formnummer	db2tug81
Lernprogramm für das Informationsmanagement: Data Warehouse - Weiterführende Informationen	Keine Formnummer	db2tag81
Lernprogramm für die Informationskatalogzentrale	Keine Formnummer	db2aig81
Video Central für e-business Lernprogramm	Keine Formnummer	db2twg81
Lernprogramm für Visual Explain	Keine Formnummer	db2tvg81

Informationen zu Zusatzkomponenten

Die Informationen in dieser Kategorie beschreiben das Arbeiten mit den DB2-Zusatzkomponenten.

Tabelle 80. Informationen zu Zusatzkomponenten

Name	IBM Form	PDF-Dateiname
IBM DB2 Cube Views Handbuch und Referenz	n/v	db2aag81
IBM DB2 Query Patroller-Handbuch: Installation, Verwaltung und Verwendung	GC12-3225	db2dwg81
IBM DB2 Spatial Extender und Geodetic Extender Benutzer- und Referenzhandbuch	SC12-3063	db2sbg81

Tabelle 80. Informationen zu Zusatzkomponenten (Forts.)

Name	IBM Form	PDF-Dateiname
IBM DB2 Universal Database Data Links Manager Administration Guide and Reference	SC27-1221	db2z0e82
DB2 Net Search Extender Verwaltung und Benutzerhandbuch **Anmerkung:** Die HTML-Version dieses Dokuments wird *nicht* von der HTML-Dokumentations-CD installiert.	SH12-3021	n/v

Release-Informationen

Die Release-Informationen enthalten zusätzliche Informationen für das verwendete Produktrelease und die verwendete FixPak-Stufe. Die Release-Informationen enthalten außerdem Zusammenfassungen der Dokumentationsaktualisierungen in den verschiedenen Releases, Aktualisierungen und FixPaks.

Tabelle 81. Release-Informationen

Name	IBM Form	PDF-Dateiname
DB2 Release-Informationen	Siehe Anmerkung.	Siehe Anmerkung.
DB2 Installationsinformationen	Nur auf der Produkt-CD-ROM verfügbar.	n/v

Anmerkung: Die Release-Informationen stehen in den folgenden Formaten zur Verfügung:

- XHTML und Textformat auf den Produkt-CDs
- PDF-Format auf der CD mit der PDF-Dokumentation

Darüber hinaus sind die Abschnitte zu *bekannten Problemen und Fehlerumgehungen* sowie zur *Inkompatibilität zwischen einzelnen Releases*, die Teil der Release-Informationen sind, auch über 'DB2 Information - Unterstützung' verfügbar.

Informationen zum Anzeigen der Release-Informationen in Textformat auf UNIX-Plattformen finden Sie in der Datei `Release.Notes`. Diese Datei befindet sich im Verzeichnis `DB2DIR/Readme/%L`. Hierbei steht `%L` für die länderspezifische Angabe und `DB2DIR` für eine der folgenden Angaben:

- Für AIX-Betriebssysteme: `/usr/opt/db2_08_01`
- Für alle anderen UNIX-Betriebssysteme: `/opt/IBM/db2/V8.1`

Zugehörige Konzepte:

- „DB2-Dokumentation und Hilfe" auf Seite 685

Zugehörige Tasks:

- „Drucken von DB2-Büchern mit PDF-Dateien" auf Seite 704
- „Bestellen gedruckter DB2-Bücher" auf Seite 704
- „Aufrufen der Kontexthilfe über ein DB2-Tool" auf Seite 705

Drucken von DB2-Büchern mit PDF-Dateien

DB2-Bücher können mit Hilfe der PDF-Dateien auf der CD mit der *DB2-PDF-Dokumentation* gedruckt werden. Mit Adobe Acrobat Reader können Sie entweder das gesamte Handbuch oder bestimmte Seitenbereiche des Handbuchs ausdrucken.

Voraussetzungen:

Stellen Sie sicher, dass Adobe Acrobat Reader installiert ist. Falls Sie Adobe Acrobat Reader noch nicht installiert haben, finden Sie das Produkt auf der Adobe-Website unter folgender Adresse: www.adobe.com

Vorgehensweise:

Gehen Sie wie folgt vor, um ein DB2-Buch mit einer PDF-Datei auszudrucken:

1. Legen Sie die CD mit der *DB2-PDF-Dokumentation* in das CD-ROM-Laufwerk ein. Hängen Sie unter UNIX-Betriebssystemen die CD mit der DB2-PDF-Dokumentation an. Informationen zum Anhängen einer CD unter UNIX-Betriebssystemen finden Sie im Handbuch *Einstieg* für das jeweilige Betriebssystem.
2. Öffnen Sie index.htm. Die Datei wird in einem Browserfenster geöffnet.
3. Klicken Sie den Titel der PDF an, die Sie aufrufen möchten. Die PDF wird in Acrobat Reader geöffnet.
4. Wählen Sie **Datei → Drucken** aus, um einen beliebigen Teil des gewünschten Buches zu drucken.

Zugehörige Konzepte:

- „DB2 Information - Unterstützung" auf Seite 686

Zugehörige Tasks:

- „Anhängen der CD-ROM (AIX)" in *DB2 Universal Database für DB2-Server Einstieg*
- „Anhängen der CD-ROM (HP-UX)" in *DB2 Universal Database für DB2-Server Einstieg*
- „Anhängen der CD-ROM (Linux)" in *DB2 Universal Database für DB2-Server Einstieg*
- „Bestellen gedruckter DB2-Bücher" auf Seite 704
- „Anhängen der CD-ROM (Solaris-Betriebsumgebung)" in *DB2 Universal Database für DB2-Server Einstieg*

Zugehörige Referenzen:

- „DB2-Dokumentation in PDF-Format und gedrucktem Format" auf Seite 698

Bestellen gedruckter DB2-Bücher

Wenn Sie die Hardcopyversion der Bücher bevorzugen, können Sie sie auf eine der nachfolgend aufgeführten Arten bestellen.

Vorgehensweise:

In bestimmten Ländern oder Regionen können gedruckte Bücher bestellt werden. Auf der Website mit IBM Veröffentlichungen für das jeweilige Land bzw. die jeweilige Region finden Sie Informationen darüber, ob dieser Service im betreffenden

Land bzw. in der betreffenden Region angeboten wird. Wenn die Veröffentlichungen bestellt werden können, haben Sie folgende Möglichkeiten:

- Wenden Sie sich an den zuständigen IBM Vertragshändler oder Vertriebsbeauftragten. Informationen zum lokalen IBM Ansprechpartner finden Sie im globalen IBM Verzeichnis für Kontakte unter folgender Adresse: www.ibm.com/planetwide.
- Weitere Informationen enthält das IBM Publications Center unter http://www.ibm.com/shop/publications/order. Die Möglichkeit, Bücher über das IBM Publications Center zu bestellen, besteht möglicherweise nicht in allen Ländern.

Die gedruckten Bücher sind zu dem Zeitpunkt, an dem das DB2-Produkt verfügbar gemacht wird, identisch mit den PDF-Versionen auf der CD mit der *DB2-PDF-Dokumentation*. Darüber hinaus stimmt der Inhalt der gedruckten Bücher mit den entsprechenden Informationen auf der CD für *DB2 Information - Unterstützung* überein. Diese CD enthält jedoch zusätzliche Informationen, die in den PDF-Büchern nicht enthalten sind (wie beispielsweise SQL-Verwaltungsroutinen und HTML-Beispiele). Nicht alle Bücher, die auf der CD mit der DB2-PDF-Dokumentation verfügbar sind, können als Hardcopy bestellt werden.

Anmerkung: 'DB2 Information - Unterstützung' wird häufiger aktualisiert als die PDF- oder die Hardcopyversion der Bücher. Installieren Sie die Dokumentationsupdates, sobald diese verfügbar sind, oder greifen Sie über 'DB2 Information - Unterstützung' unter http://publib.boulder.ibm.com/infocenter/db2help/ auf die neuesten Informationen zu.

Zugehörige Tasks:

- „Drucken von DB2-Büchern mit PDF-Dateien" auf Seite 704

Zugehörige Referenzen:

- „DB2-Dokumentation in PDF-Format und gedrucktem Format" auf Seite 698

Aufrufen der Kontexthilfe über ein DB2-Tool

Die Kontexthilfe bietet Informationen zu den Tasks bzw. Steuerelementen, die einem bestimmten Fenster, Notizbuch, Assistenten oder Advisor zugeordnet sind. Die Kontexthilfe steht in allen DB2-Verwaltungs- und -entwicklungstools zur Verfügung, die über eine grafische Benutzerschnittstelle verfügen. Zwei Arten der Kontexthilfe stehen zur Verfügung:

- Die über den Knopf **Hilfe** aufgerufenen Hilfetexte, der in jedem Fenster bzw. Notizbuch zur Verfügung steht.
- Die Kurzhilfe. Hierbei handelt es sich um Informationsfenster, die angezeigt werden, wenn sich der Mauszeiger auf einem Feld oder Steuerelement befindet oder wenn bei der Auswahl eines Feldes oder Steuerelements in einem Fenster, Notizbuch, Assistenten oder Advisor die Taste F1 gedrückt wird.

Über den Knopf **Hilfe** können Sie auf Übersichtsinformationen, Informationen zu Voraussetzungen sowie Informationen zu Tasks zugreifen. In der Kurzhilfe werden die einzelnen Felder und Steuerelemente beschrieben.

Vorgehensweise:

Gehen Sie wie folgt vor, um Kontexthilfe aufzurufen:

- Hilfe zu Fenstern und Notizbüchern können Sie anzeigen, indem Sie eines der DB2-Tools aufrufen und anschließend ein beliebiges Fenster oder Notizbuch öffnen. Klicken Sie den Knopf **Hilfe** in der rechten unteren Ecke des Fensters bzw. Notizbuchs an, um die Kontexthilfe aufzurufen.

  Zugriff auf die Kontexthilfe besteht darüber hinaus über den Menüpunkt **Hilfe** am oberen Rand jeder Zentrale der DB2-Tools.

  Innerhalb von Assistenten und Advisorfunktionen klicken Sie den Link für die Taskübersicht auf der ersten Seite an, um die Kontexthilfe aufzurufen.
- Kurzhilfe zu einzelnen Steuerelementen eines Fensters oder Notizbuchs können Sie aufrufen, indem Sie das gewünschte Steuerelement anklicken und anschließend **F1** drücken. Die Kurzhilfeinformationen mit Details zum jeweiligen Steuerelement werden in einem gelben Fenster angezeigt.

  Anmerkung: Wenn die Kurzhilfe angezeigt werden soll, sobald sich der Mauszeiger auf einem Feld oder Steuerelement befindet, wählen Sie das Markierungsfeld **Kurzhilfe automatisch anzeigen** auf der Seite **Dokumentation** des Notizbuchs 'Tools - Einstellungen' aus.

  Ähnlich wie die Kurzhilfe sind auch Dialogfenster mit Diagnoseinformationen eine Form der kontextbezogenen Hilfe; sie enthalten Regeln für die Dateneingabe. Diese Diagnoseinformationen werden in einem violetten Fenster angezeigt, das aufgerufen wird, wenn die eingegebenen Daten nicht gültig oder nicht ausreichend sind. Die Kontexthilfe mit Diagnoseinformationen kann für folgende Felder angezeigt werden:
  - Musseingabefelder
  - Felder, in denen die Daten einem bestimmten Format entsprechen müssen, wie z. B. Datumsfelder

Zugehörige Tasks:

- „Aufrufen von 'DB2 Information - Unterstützung'" auf Seite 695
- „Aufrufen der Hilfe für Nachrichten über den Befehlszeilenprozessor" auf Seite 706
- „Aufrufen der Hilfe für Befehle über den Befehlszeilenprozessor" auf Seite 707
- „Aufrufen der Hilfe für den SQL-Status über den Befehlszeilenprozessor" auf Seite 707
- „Zugriff auf 'DB2 Information - Unterstützung': Konzepthilfe"
- „Verwenden der DB2 UDB-Hilfe: Gemeinsame GUI - Hilfe"
- „Festlegen der Speicherposition für den Zugriff auf 'DB2 Information - Unterstützung': Gemeinsame GUI - Hilfe"
- „Einrichten des Zugriffs auf DB2-Kontexthilfe und -Dokumentation: Gemeinsame GUI - Hilfe"

Aufrufen der Hilfe für Nachrichten über den Befehlszeilenprozessor

Die Hilfe für Nachrichten beschreibt die Ursache von Nachrichten und die Aktionen, die der Benutzer zur Behebung des aufgetretenen Fehlers ausführen sollte.

Vorgehensweise:

Zum Aufrufen der Hilfe für Nachrichten müssen Sie den Befehlszeilenprozessor öffnen und Folgendes eingeben:

```
? XXXnnnnn
```

Dabei ist *XXXnnnnn* eine gültige Nachrichtenkennung.

So kann beispielsweise durch die Eingabe von `? SQL30081` die Hilfe zur Nachricht SQL30081 angezeigt werden.

Zugehörige Konzepte:

- „Nachrichten - Einführung" in *Fehlernachrichten Band 1*

Zugehörige Referenzen:

- „db2 - Command Line Processor Invocation Command" in *Command Reference*

Aufrufen der Hilfe für Befehle über den Befehlszeilenprozessor

Die Hilfe für Befehle erläutert die Syntax von Befehlen im Befehlszeilenprozessor.

Vorgehensweise:

Zum Aufrufen der Hilfe für Befehle müssen Sie den Befehlszeilenprozessor öffnen und Folgendes eingeben:

```
? command
```

Dabei stellt *command* ein Schlüsselwort bzw. den vollständigen Befehl dar.

So kann beispielsweise durch die Eingabe von `? catalog` Hilfe für alle CATALOG-Befehle angezeigt werden, während mit `? catalog database` nur Hilfe für den Befehl CATALOG DATABASE angezeigt wird.

Zugehörige Tasks:

- „Aufrufen der Kontexthilfe über ein DB2-Tool" auf Seite 705
- „Aufrufen von 'DB2 Information - Unterstützung'" auf Seite 695
- „Aufrufen der Hilfe für Nachrichten über den Befehlszeilenprozessor" auf Seite 706
- „Aufrufen der Hilfe für den SQL-Status über den Befehlszeilenprozessor" auf Seite 707

Zugehörige Referenzen:

- „db2 - Command Line Processor Invocation Command" in *Command Reference*

Aufrufen der Hilfe für den SQL-Status über den Befehlszeilenprozessor

DB2 Universal Database gibt für Bedingungen, die auf Grund 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 Aufrufen der Hilfe für SQL-Statuswerte müssen Sie den Befehlszeilenprozessor öffnen und Folgendes eingeben:

```
? sqlstate oder ? klassencode
```

Hierbei steht *sqlstate* für einen gültigen fünfstelligen SQL-Statuswert und *klassencode* für die ersten beiden Ziffern dieses Statuswertes.

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.

Zugehörige Tasks:

- „Aufrufen von 'DB2 Information - Unterstützung'" auf Seite 695
- „Aufrufen der Hilfe für Nachrichten über den Befehlszeilenprozessor" auf Seite 706
- „Aufrufen der Hilfe für Befehle über den Befehlszeilenprozessor" auf Seite 707

DB2-Lernprogramme

Die Lernprogramme von DB2® unterstützen Sie bei der Einarbeitung in die verschiedenen Themenbereiche von DB2 Universal Database. Sie umfassen Übungen mit in einzelne Arbeitsschritte untergliederten Anweisungen zum Entwickeln von Anwendungen, Optimieren der SQL-Abfrageleistung, Arbeiten mit Data Warehouses, Verwalten von Metadaten und Entwickeln von Webservices mit Hilfe von DB2.

Vorbereitungen:

Die XHTML-Version der Lernprogramme kann über 'DB2 Information - Unterstützung' unter http://publib.boulder.ibm.com/infocenter/db2help/ angezeigt werden.

In einigen der Lernprogrammübungen werden Beispieldaten und Codebeispiele verwendet. Informationen zu den spezifischen Voraussetzungen zur Ausführung der Tasks finden Sie in der Beschreibung des jeweiligen Lernprogramms.

Lernprogramme von DB2 Universal Database:

Klicken Sie einen der Lernprogrammtitel in der folgenden Liste an, um das entsprechende Lernprogramm aufzurufen.

Lernprogramm für das Informationsmanagement: Data Warehouse - Einführung
: Ausführung grundlegender Data Warehousing-Tasks mit Hilfe der Data Warehouse-Zentrale.

Lernprogramm für das Informationsmanagement: Data Warehouse - Weiterführende Informationen
: Ausführung weiterführender Data Warehousing-Tasks mit Hilfe der Data Warehouse-Zentrale.

Lernprogramm für die Informationskatalogzentrale
: Erstellen und Verwalten eines Informationskatalogs zum Lokalisieren und Verwenden von Metadaten mit Hilfe der Informationskatalogzentrale.

Lernprogramm für Visual Explain
: Analysieren, Optimieren und Anpassen von SQL-Anweisungen zur Leistungsverbesserung mit Hilfe von Visual Explain.

Informationen zur Fehlerbehebung in DB2

Eine breite Palette verschiedener Informationen zur Fehlerbestimmung und Fehlerbehebung steht zur Verfügung, um Sie bei der Verwendung von DB2®-Produkten zu unterstützen.

DB2-Dokumentation

Informationen zur Fehlerbehebung stehen in der gesamten Komponente 'DB2 Information - Unterstützung' sowie in den PDF-Büchern der DB2-Bibliothek zur Verfügung. Folgen Sie der Verzweigung 'Unterstützung und Fehlerbehebung' in der Navigationsbaumstruktur von 'DB2 Information - Unterstützung' (im linken Teilfenster des Browserfensters), um eine umfassende Liste der DB2-Dokumentationen zur Fehlerbehebung aufzurufen.

DB2-Website mit technischer Unterstützung

Auf der DB2-Website mit technischer Unterstützung finden Sie Informationen zu Problemen und den möglichen Ursachen und Fehlerbehebungsmaßnahmen. Die DB2-Website mit technischer Unterstützung stellt Links zu den neuesten DB2-Veröffentlichungen, technischen Hinweisen (TechNotes), APARs (Authorized Program Analysis Reports), FixPaks, den neuesten Listen mit internen DB2-Fehlercodes sowie weiteren Ressourcen zur Verfügung. Sie können diese Wissensbasis nach möglichen Lösungen für aufgetretene Probleme durchsuchen.

Rufen Sie die DB2-Website mit technischer Unterstützung unter http://www.ibm.com/software/data/db2/udb/winos2unix/support auf.

DB2-Lernprogramme zur Fehlerbestimmung

Auf der Website mit den DB2-Lernprogrammen zur Fehlerbestimmung finden Sie Informationen dazu, wie Sie Fehler, die bei der Verwendung von DB2-Produkten möglicherweise auftreten, rasch identifizieren und beheben können. Eines der Lernprogramme bietet eine Einführung in die verfügbaren DB2-Einrichtungen und -Tools zur Fehlerbestimmung sowie Entscheidungshilfen für deren Verwendung. Andere Lernprogramme befassen sich mit zugehörigen Themen, wie beispielsweise der Fehlerbestimmung für die Datenbanksteuerkomponente, der Fehlerbestimmung für die Leistung und der Fehlerbestimmung für Anwendungen.

Die vollständige Liste der DB2-Lernprogramme zur Fehlerbestimmung finden Sie auf der DB2-Website mit technischer Unterstützung unter http://www.ibm.com/software/data/support/pdm/db2tutorials.html.

Zugehörige Konzepte:

- „DB2 Information - Unterstützung" auf Seite 686
- „Einführung in die Fehlerbestimmung - Lernprogramm für die technische Unterstützung in DB2" in *Troubleshooting Guide*

Eingabehilfen

Eingabehilfen unterstützen Benutzer mit körperlichen Behinderungen, wie z. B. eingeschränkter Bewegungsfähigkeit oder Sehkraft, beim erfolgreichen Einsatz von Softwareprodukten. Im Folgenden sind die wichtigsten Eingabehilfen aufgeführt, die in den Produkten von DB2® Version 8 zur Verfügung stehen:

- Die gesamte DB2-Funktionalität kann sowohl über die Maus als auch über die Tastatur gesteuert werden. Weitere Informationen hierzu finden Sie unter „Tastatureingabe und Navigation" auf Seite 710.
- Sie können die Größe und Farbe der verwendeten Schriftarten in den DB2-Schnittstellen anpassen. Weitere Informationen hierzu finden Sie unter „Eingabehilfen für Bildschirme" auf Seite 710.
- DB2-Produkte unterstützen Anwendungen mit Eingabehilfen, die mit der Java™ Accessibility API arbeiten. Weitere Informationen hierzu finden Sie unter „Kompatibilität mit Unterstützungseinrichtungen" auf Seite 710.

- Die DB2-Dokumentation steht in behindertengerechtem Format zur Verfügung. Weitere Informationen hierzu finden Sie unter „Dokumentation im behindertengerechten Format".

Tastatureingabe und Navigation

Tastatureingabe

Die verfügbaren DB2-Tools können unter ausschließlicher Benutzung der Tastatur verwendet werden. Mit entsprechenden Tasten oder Tastenkombinationen können Operationen ausgeführt werden, die auch über die Maus verfügbar sind. Die Standardtastenkombinationen des Betriebssystems werden für die entsprechenden Standardoperationen des Betriebssystems verwendet.

Weitere Informationen zur Verwendung von Tasten oder Tastenkombinationen für die Ausführung von Operationen finden Sie unter '' 'Direktaufrufe über die Tastatur: Gemeinsame GUI - Hilfe'.

Navigation über die Tastatureingabe

Sie können in den Benutzerschnittstellen der DB2-Tools mit Hilfe von Tasten oder Tastenkombinationen navigieren.

Weitere Informationen zur Navigation in den DB2-Tools mit Hilfe der Tastatureingabe finden Sie unter '' 'Direktaufrufe über die Tastatur: Gemeinsame GUI - Hilfe'.

Tastatureingabebereich

Unter UNIX®-Betriebssystemen ist der Bereich des aktiven Fensters, in dem die Tastatureingabe wirksam ist, hervorgehoben.

Eingabehilfen für Bildschirme

Die DB2-Tools stellen Funktionen bereit, mit denen sehbehinderten Benutzern verbesserten Eingabehilfen zur Verfügung stehen. Diese Eingabehilfen umfassen die Unterstützung individuell anpassbarer Schriftarteigenschaften.

Schriftarteinstellungen

Über das Notizbuch 'Tools - Einstellungen' können Sie die Farbe, Größe und Schriftart des Textes in Menüs und Dialogfenstern auswählen.

Weitere Informationen zur Angabe von Schriftarteinstellungen finden Sie unter '' 'Ändern der Schriftarten für Menüs und Text: Gemeinsame GUI - Hilfe'.

Unabhängigkeit von Farben

Zur Verwendung der Funktionen des vorliegenden Produkts ist es nicht erforderlich, zwischen unterschiedlichen Farben differenzieren zu können.

Kompatibilität mit Unterstützungseinrichtungen

Die Schnittstellen der DB2-Tools unterstützen die Java Accessibility API. Hierdurch wird der Einsatz von Sprachausgabeprogrammen und anderen Unterstützungseinrichtungen für Personen mit Behinderungen mit den DB2-Produkten ermöglicht.

Dokumentation im behindertengerechten Format

Die Dokumentation für DB2 steht im Format XHTML 1.0 zur Verfügung, das mit den meisten Webbrowsern geöffnet werden kann. XHTML ermöglicht das Aufrufen der Dokumentation mit den Anzeigeeinstellungen, die Sie in Ihrem Browser definiert haben. Darüber hinaus ist der Einsatz von Sprachausgabeprogrammen und anderen Unterstützungseinrichtungen möglich.

Syntaxdiagramme stehen in der Schreibweise mit Trennzeichen zur Verfügung. Dieses Format ist nur dann verfügbar, wenn Sie mit Hilfe eines Sprachausgabeprogramms auf die Onlinedokumentation zugreifen.

Zugehörige Konzepte:

- „Syntaxdiagramme in der Schreibweise mit Trennzeichen" auf Seite 711

Zugehörige Tasks:

- „Direktaufrufe über die Tastatur: Gemeinsame GUI - Hilfe"
- „Ändern der Schriftarten für Menüs und Text: Gemeinsame GUI - Hilfe"

Syntaxdiagramme in der Schreibweise mit Trennzeichen

Syntaxdiagramme stehen für Benutzer, die mit Hilfe eines Sprachausgabeprogramms auf 'DB2 Information - Unterstützung' zugreifen, in der Schreibweise mit Trennzeichen zur Verfügung.

In der Schreibweise mit Trennzeichen steht jedes Syntaxelement in einer separaten Zeile. Wenn zwei oder mehr Syntaxelemente stets gemeinsam angegeben (oder nicht angegeben) werden müssen, können sie in derselben Zeile stehen, da sie als ein zusammengesetztes Syntaxelement betrachtet werden können.

Jede Zeile beginnt mit einer Zahl in der Schreibweise mit Trennzeichen, zum Beispiel 3 oder 3.1 oder 3.1.1. Um diese Zahlen korrekt zu hören, müssen Sie sicherstellen, dass das Sprachausgabeprogramm so konfiguriert ist, dass die Interpunktion angesagt wird. Alle Syntaxelemente mit derselben Zahl in der Schreibweise mit Trennzeichen (z. B. alle Syntaxelemente mit der Zahl 3.1) stellen Alternativen dar, die sich gegenseitig ausschließen. Wenn Sie die Zeilen '3.1 USERID' und '3.1 SYSTEMID' hören, wissen Sie, dass die Syntax entweder USERID oder SYSTEMID enthalten kann, nicht jedoch beides.

Die Nummerierung bei der Schreibweise mit Trennzeichen gibt den Grad der Ausgliederung an. Beispiel: Wenn auf das Syntaxelement mit der Zahl 3 in der Schreibweise mit Trennzeichen eine Reihe von Syntaxelementen mit der Zahl 3.1 folgt, sind alle Syntaxelemente mit der Zahl 3.1 dem Syntaxelement mit der Zahl 3 untergeordnet.

Bestimmte Wörter und Symbole werden zusätzlich zu den Zahlen in der Schreibweise mit Trennzeichen verwendet, um weitere Informationen zu den Syntaxelementen anzugeben. In manchen Fällen können diese Wörter und Symbole am Anfang des Elements selbst stehen. Zur einfacheren Identifizierung wird dem Wort oder Symbol ein umgekehrter Schrägstrich (\) vorangestellt, wenn es Teil des Syntaxelements ist. Das Symbol * (Stern) kann zusätzlich zu einer Zahl in der Schreibweise mit Trennzeichen verwendet werden, um anzugeben, dass das Syntaxelement wiederholt wird. Beispiel: Das Syntaxelement *FILE mit der Zahl 3 in der Schreibweise mit Trennzeichen erhält das Format 3 * FILE. Format 3* FILE gibt an, dass das Syntaxelement FILE wiederholt wird. Format 3* * FILE gibt an, dass das Syntaxelement * FILE wiederholt wird.

Zeichen wie beispielsweise Kommas, die bei einer Folge von Syntaxelementen als Trennzeichen verwendet werden, werden in der Syntax unmittelbar vor den Elementen dargestellt, die sie trennen. Diese Zeichen können in derselben Zeile stehen wie das jeweilige Element oder in einer separaten Zeile mit derselben Zahl in der Schreibweise mit Trennzeichen, die auch dem betreffenden Element zugeordnet ist. Die Zeile kann auch ein weiteres Symbol enthalten, das Informationen zu den

Syntaxelementen angibt. So bedeuten z. B. die Zeilen 5.1*, 5.1 LASTRUN und 5.1 DELETE, dass, wenn Sie mehr als eines der Elemente LASTRUN und DELETE verwenden, diese Elemente durch Kommas voneinander getrennt werden müssen. Wenn kein Trennzeichen angegeben wird, verwendet das System zum Trennen der einzelnen Syntaxelemente ein Leerzeichen.

Wenn einem Syntaxelement das Symbol % vorangestellt ist, gibt dies einen Verweis an, der an anderer Stelle definiert ist. Die Zeichenfolge, die auf das Symbol % folgt, ist der Name eines Syntaxfragments und kein Literal. So gibt die Zeile 2.1 %OP1 beispielsweise einen Verweis auf das separate Syntaxfragment OP1 an.

Die nachfolgend aufgeführten Wörter und Symbole werden zusätzlich zu den Zahlen in der Schreibweise mit Trennzeichen verwendet:

- ? stellt ein optionales Syntaxelement dar. Eine Zahl in der Schreibweise mit Trennzeichen, auf die das Symbol ? (Fragezeichen) folgt, gibt an, dass alle Syntaxelemente mit einer entsprechenden Zahl in der Schreibweise mit Trennzeichen sowie alle untergeordneten Syntaxelemente optional sind. Ist nur ein Syntaxelement mit einer Zahl in der Schreibweise mit Trennzeichen vorhanden, wird das Symbol ? in derselben Zeile angezeigt wie das Syntaxelement (zum Beispiel 5? NOTIFY). Sind mehrere Syntaxelemente mit einer Zahl in der Schreibweise mit Trennzeichen vorhanden, wird das Symbol ? in einer separaten Zeile angezeigt, gefolgt von den optionalen Syntaxelementen. Wenn Sie beispielsweise die Zeilen 5 ?, 5 NOTIFY und 5 UPDATE hören, wissen Sie, dass die Syntaxelemente NOTIFY und UPDATE optional sind; das bedeutet, Sie können eines oder keines dieser Elemente auswählen. Das Symbol ? entspricht einer Umgehungslinie in einem Pfeildiagramm.
- ! stellt ein Standardsyntaxelement dar. Eine Zahl in der Schreibweise mit Trennzeichen, auf die das Symbol ! (Ausrufezeichen) und ein Syntaxelement folgen, gibt an, dass es sich bei diesem Syntaxelement um die Standardoption für alle Syntaxelemente handelt, denen dieselbe Zahl in der Schreibweise mit Trennzeichen zugeordnet ist. Nur für eines der Syntaxelemente, denen dieselbe Zahl in der Schreibweise mit Trennzeichen zugeordnet ist, darf das Symbol ! angegeben werden. Wenn Sie beispielsweise die Zeilen 2? FILE, 2.1! (KEEP) und 2.1 (DELETE) hören, wissen Sie, dass (KEEP) die Standardoption für das Schlüsselwort FILE ist. Wenn Sie in diesem Beispiel das Schlüsselwort FILE verwenden, jedoch keine Option angeben, wird die Standardoption KEEP verwendet. Eine Standardoption ist auch für die nächsthöhere Zahl in der Schreibweise mit Trennzeichen gültig. In diesem Beispiel bedeutet das: Wenn das Schlüsselwort FILE weggelassen wird, wird der Standardwert FILE(KEEP) verwendet. Wenn Sie jedoch die Zeilen 2? FILE, 2.1, 2.1.1! (KEEP) und 2.1.1 (DELETE) hören, gilt die Standardoption KEEP nur für die nächsthöhere Zahl in der Schreibweise mit Trennzeichen, 2.1 (der kein Schlüsselwort zugeordnet ist), nicht jedoch für 2? FILE. Wird das Schlüsselwort FILE weggelassen, wird kein Wert verwendet.
- * stellt ein Syntaxelement dar, das keinmal, einmal oder mehrmals wiederholt werden kann. Eine Zahl in der Schreibweise mit Trennzeichen, auf die das Symbol * (Stern) folgt, gibt an, dass dieses Syntaxelement keinmal, einmal oder mehrmals verwendet werden kann, d. h., es ist optional und kann wiederholt werden. Wenn Sie beispielsweise die Zeile 5.1* Datenbereich hören, wissen Sie, dass Sie einen, mehrere oder keinen Datenbereich angeben können. Hören Sie die Zeilen 3*, 3 HOST und 3 STATE, wissen Sie, dass Sie HOST, STATE, beide oder keines der Elemente angeben können.

Anmerkungen:

1. Wenn neben einer Zahl in der Schreibweise mit Trennzeichen ein Stern (*) angezeigt wird und nur ein Element mit dieser Zahl vorhanden ist, können Sie dieses Element mehrmals wiederholen.
2. Wenn neben einer Zahl in der Schreibweise mit Trennzeichen ein Stern angezeigt wird und diese Zahl mehreren Elementen zugeordnet ist, können Sie mehrere Elemente aus der Liste verwenden, jedes davon jedoch nur einmal. Im vorhergehenden Beispiel könnten Sie HOST STATE angeben, nicht jedoch HOST HOST.
3. Das Symbol * entspricht einer zum Ausgangspunkt zurück führenden Linie in einem Pfeildiagramm.

- + stellt ein Syntaxelement dar, das mindestens einmal angegeben werden muss. Eine Zahl in der Schreibweise mit Trennzeichen, auf die das Symbol + (Pluszeichen) folgt, gibt an, dass dieses Syntaxelement mindestens einmal angegeben werden muss und wiederholt werden kann. Wenn Sie beispielsweise die Zeile 6.1+ Datenbereich hören, müssen sie mindestens einen Datenbereich angeben. Wenn Sie die Zeilen 2+, 2 HOST und 2 STATE hören, wissen Sie, dass Sie HOST, STATE oder beides angeben müssen. Wie auch für das Symbol * gilt hier, dass mit dem Pluszeichen ein bestimmtes Element nur dann wiederholt werden kann, wenn es sich um das einzige Element mit dieser Zahl in der Schreibweise mit Trennzeichen handelt. Das Symbol + entspricht wie das Symbol * einer zum Ausgangspunkt zurück führenden Linie in einem Pfeildiagramm.

Zugehörige Konzepte:

- „Eingabehilfen" auf Seite 709

Zugehörige Tasks:

- „Direktaufrufe über die Tastatur: Gemeinsame GUI - Hilfe"

Zugehörige Referenzen:

- „How to read the syntax diagrams" in *SQL Reference, Volume 2*

Common Criteria-Zertifizierung von DB2 Universal Database-Produkten

Für DB2 Universal Database läuft momentan der Bewertungsprozess für die Zertifizierung entsprechend den Richtlinien von Common Criteria Evaluation Assurance Level 4 (EAL4). Weitere Informationen zu Common Criteria finden Sie auf der Common Criteria-Website unter: http://niap.nist.gov/cc-scheme/.

Anhang G. Bemerkungen

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 IBM Ansprechpartner erhältlich. Hinweise auf IBM Lizenzprogramme oder andere IBM Produkte bedeuten nicht, dass nur Programme, Produkte oder Dienstleistungen von IBM verwendet werden können. An Stelle der IBM Produkte, Programme oder Dienstleistungen können auch andere ihnen äquivalente Produkte, Programme oder Dienstleistungen verwendet werden, solange diese keine gewerblichen oder anderen Schutzrechte der IBM verletzen. Die Verantwortung für den Betrieb der Produkte, Programme oder Dienstleistungen in Verbindung mit Fremdprodukten und Fremddienstleistungen liegt beim Kunden, soweit nicht ausdrücklich solche Verbindungen erwähnt sind.

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 Europe, Director of Licensing, 92066 Paris La Defense Cedex, France.

Trotz sorgfältiger Bearbeitung können technische Ungenauigkeiten oder Druckfehler in dieser Veröffentlichung nicht ausgeschlossen werden. Die Angaben in diesem Handbuch werden in regelmäßigen Zeitabständen aktualisiert. Die Änderungen werden in Überarbeitungen bekanntgegeben. IBM kann jederzeit Verbesserungen und/oder Änderungen an den in dieser Veröffentlichung beschriebenen Produkten und/oder Programmen vornehmen.

Verweise in diesen Informationen auf Websites anderer Anbieter dienen lediglich als Benutzerinformationen 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 unabhängigen, 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
Office of the Lab Director
8200 Warden Avenue
Markham, Ontario
L6G 1C7
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 Handbuch aufgeführten Lizenzprogramms sowie des zugehörigen Lizenzmaterials erfolgt im Rahmen der Allgemeinen Geschäftsbedingungen der IBM, der Internationalen Nutzungsbedingungen der IBM für Programmpakete oder einer äquivalenten Vereinbarung.

Alle in diesem Dokument enthaltenen Leistungsdaten stammen aus einer gesteuerten 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 Garantie, 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 abweichen. Benutzer dieses Dokuments sollten die entsprechenden Daten in ihrer spezifischen Umgebung prüfen.

Informationen über Produkte anderer Hersteller als IBM wurden von den Herstellern dieser Produkte zur Verfügung gestellt, bzw. aus von ihnen veröffentlichten Ankündigungen oder anderen öffentlich zugänglichen Quellen entnommen. IBM hat diese Produkte nicht getestet und übernimmt im Hinblick auf Produkte anderer Hersteller keine Verantwortung für einwandfreie Funktion, Kompatibilität oder andere Ansprüche. Fragen zu den Leistungsmerkmalen von Produkten anderer Anbieter sind an den jeweiligen Anbieter zu richten.

Aussagen über Pläne und Absichten der IBM unterliegen Änderungen oder können zurückgenommen werden und repräsentieren nur die Ziele der IBM.

Diese Veröffentlichung enthält Beispiele für Daten und Berichte des alltäglichen Geschäftsablaufes. Sie sollen nur die Funktionen des Lizenzprogrammes 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 Beispielanwendungsprogramme, die in Quellensprache geschrieben sind. Sie dürfen diese Beispielprogramme kostenlos kopieren, ändern und verteilen, wenn dies zu dem Zweck geschieht, Anwendungsprogramme zu entwickeln, verwenden, vermarkten oder zu verteilen, die mit der Anwendungsprogrammierschnittstelle konform sind, für die diese Beispielprogramme 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.

Kopien oder Teile der Beispielprogramme bzw. daraus abgeleiteter Code müssen folgenden Copyrightvermerk beinhalten:

© (*Name Ihrer Firma*) (*Jahr*). Teile des vorliegenden Codes wurden aus Beispielprogrammen der IBM Corp. abgeleitet. © Copyright IBM Corp. *_Jahr/Jahre angeben_*. Alle Rechte vorbehalten.

Marken

Folgende Namen sind in gewissen Ländern Marken der International Business Machines Corporation und wurden in mindestens einem der Dokumente in der DB2 UDB-Dokumentationsbibliothek verwendet:

ACF/VTAM
AISPO
AIX
AIXwindows
AnyNet
APPN
IBM System AS/400
BookManager
C Set++
C/370
CICS
Database 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extenders
DB2 OLAP Server
DB2 Information Integrator
DB2 Query Patroller
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eServer
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
iSeries
LAN Distance
MVS
MVS/ESA
MVS/XA
Net.Data
NetView
OS/390
OS/400
PowerPC
pSeries
QBIC
QMF
RACF
RS/6000
S/370
SP
SQL/400
SQL/DS
System/370
IBM System /390
SystemView
Tivoli
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WebSphere
WIN-OS/2
z/OS
zSeries

Folgende Namen sind in gewissen Ländern Marken oder eingetragene Marken anderer Unternehmen und wurden in mindestens einem der Dokumente in der DB2 UDB-Dokumentationsbibliothek verwendet.

Microsoft, Windows, Windows NT und das Windows-Logo sind in gewissen Ländern Marken der Microsoft Corporation.

Intel und Pentium sind in gewissen Ländern Marken der Intel Corporation.

Java und alle auf Java basierenden Marken sind in gewissen Ländern Marken von Sun Microsystems, Inc.

UNIX ist in gewissen Ländern eine eingetragene Marke von The Open Group.

Andere Namen von Unternehmen, Produkten oder Dienstleistungen können Marken anderer Unternehmen sein.

Index

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

Kontaktaufnahme mit IBM

Telefonische Unterstützung erhalten Sie über folgende Nummern:

- Unter 0180 3 313233 erreichen Sie Hallo IBM, wo Sie Antworten zu allgemeinen Fragen erhalten.
- Unter 0190 7 72243 erreichen Sie die DB2 Helpline, wo Sie Antworten zu DB2-spezifischen Problemen erhalten.

Informationen zur nächsten IBM Niederlassung in Ihrem Land oder Ihrer Region finden Sie im IBM Verzeichnis für weltweite Kontakte, das Sie im Web unter http://www.ibm.com/planetwide abrufen können.

Produktinformationen

Informationen zu DB2 Universal Database-Produkten erhalten Sie telefonisch oder im World Wide Web unter http://www.ibm.com/software/data/db2/udb.

Diese Site enthält die neuesten Informationen zur technischen Bibliothek, zum Bestellen von Büchern, zu Produktdownloads, Newsgroups, FixPaks, Neuerungen und Links auf verfügbare Webressourcen.

Telefonische Unterstützung erhalten Sie über folgende Nummern:

- Unter 0180 3 313233 erreichen Sie Hallo IBM, wo Sie Antworten zu allgemeinen Fragen erhalten.
- Unter 0180 5 5090 können Sie Handbücher telefonisch bestellen.

Informationen dazu, wie Sie sich mit IBM in Verbindung setzen können, finden Sie auf der globalen IBM Internet-Seite unter folgender Adresse:
www.ibm.com/planetwide

SC12-3058-01