IBM DB2 Universal Database

Systemverwaltung: Konzept

Version 8.2

SC12-3057-01

IBM DB2 Universal Database

Systemverwaltung: Konzept

Version 8.2

SC12-3057-01

Hinweis

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 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: Planning Version 8.2,
IBM Form SC09-4822-01,
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. **Visual Explain** und **Performance Monitor** werden von der Steuerzentrale aus 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:

Datenbankkonzepte

- Kapitel 1, „Allgemeine Konzepte relationaler Datenbanken", enthält eine Übersicht über Datenbankobjekte und Datenbankkonzepte.
- Kapitel 2, „Parallele Datenbanksysteme", enthält eine Einführung in die Arten von Parallelität, die mit Hilfe von DB2 implementiert werden können.
- Kapitel 3, „Data Warehouses", enthält eine Übersicht über den Einsatz von Data Warehouses und Data Warehouse-Funktionen.

Datenbankentwurf

- Kapitel 4, „Logischer Datenbankentwurf", behandelt die Konzepte und Richtlinien zum logischen Entwurf einer Datenbank.
- Kapitel 5, „Physischer Datenbankentwurf", behandelt die Richtlinien für den physischen Entwurf einer Datenbank, einschließlich des Speicherbedarfs und des Entwurfs von Tabellenbereichen.
- Kapitel 6, „Entwerfen verteilter Datenbanken", beschreibt die Möglichkeit des Zugriffs auf mehrere Datenbanken in einer einzigen Transaktion.
- Kapitel 7, „Entwerfen für XA-konforme Transaktionsmanager", behandelt die Verwendung von Datenbanken in einer Umgebung für verteilte Transaktionsverarbeitung.

Anhänge

- Anhang A, „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.
- Anhang B, „Unterstützung von Landessprachen (NLS)", gibt eine Einführung in die DB2-Unterstützung von Landessprachen und enthält Informationen zu Gebieten, Sprachen und Codepages.
- Anhang C, „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.

Kurzübersicht über die anderen Bände des Handbuchs zur Systemverwaltung

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, die vor der Erstellung einer Datenbank erfüllt sein müssen.
- 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 Benennungsregeln enthält die Regeln, die bei der Benennung von Datenbanken und Objekten zu beachten sind.
- Der Anhang zu den LDAP-Verzeichnisservices enthält Informationen zu den Einsatzmöglichkeiten der LDAP-Verzeichnisservices.

- Der Anhang über das Absetzen von Befehlen an mehrere Datenbankpartitionsserver 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 über die Arbeitsweise von DB2 für Windows NT mit der Windows NT-Sicherheit beschreibt, wie DB2 mit der Windows NT-Sicherheit funktioniert.
- 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 Arbeit mit 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*.

Systemverwaltung: Optimierung

Der Band *Systemverwaltung: Optimierung* behandelt Themen zur Systemleistung. Damit sind die Themen gemeint, die das Einrichten, Testen und Optimieren der Leistung von Anwendungen sowie der Leistung des Produkts DB2 Universal Database betreffen. Die einzelnen Kapitel und Anhänge des Bandes werden im Folgenden kurz vorgestellt:

Einführung in die Optimierung

- Das Kapitel ″Einführung in die Optimierung″ führt in die Konzepte und Überlegungen zur Verwaltung und Optimierung der Leistung von DB2 UDB ein.
- Das Kapitel ″Architektur und Prozesse″ stellt die zugrunde liegende Architektur und die Prozesse von DB2 Universal Database vor.

Optimieren der Anwendungsleistung

- Die Überlegungen zu den Anwendungen beschreiben einige Techniken zur Verbesserung der Datenbankleistung beim Entwurf der Anwendungen.
- Die Überlegungen zur Umgebung beschreiben einige Techniken zur Verbesserung der Datenbankleistung bei der Einrichtung der Datenbankumgebung.
- Das Kapitel zu den Systemkatalogstatistiken beschreibt einige Techniken zur Erfassung von Statistiken über die Daten und ihre Verwendung zur Gewährleistung einer optimalen Leistung.
- Das Kapitel zum SQL-Compiler beschreibt die Verarbeitung einer SQL-Anweisung durch den SQL-Compiler bei der Kompilierung.
- Das Kapitel zur 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

- Das Kapitel zur 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.
- Das Kapitel zum Programm Governor gibt eine Einführung in die Verwendung des Programms Governor, mit dem einige Aspekte der Datenbankverwaltung gesteuert werden können.
- Das Kapitel zum Skalieren Ihrer Konfiguration enthält einige Informationen und Hinweise, die bei der Erweiterung des Datenbanksystems von Bedeutung sind.
- Das Kapitel zur Umverteilung von Daten auf Datenbankpartitionen behandelt die Punkte, die in Umgebungen mit partitionierten Datenbanken bei der Neuverteilung von Daten auf die Partitionen zu beachten sind.
- Das Kapitel zu Vergleichstests (Benchmark-Tests) enthält eine Übersicht über Vergleichstests und behandelt verschiedene Aspekte ihrer Durchführung.
- Das Kapitel zur DB2-Konfiguration 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

- Der Anhang zu DB2-Registriervariablen und DB2-Umgebungsvariablen beschreibt Werte für die Profilregistrierdatenbank und für Umgebungsvariablen.
- Der Anhang zu EXPLAIN-Tabellen und -Definitionen beschreibt die Tabellen, die von der DB2-EXPLAIN-Einrichtung verwendet werden, und die Erstellung dieser Tabellen.
- Der Anhang zu den SQL-Explain-Programmen beschreibt die Verwendung der DB2-EXPLAIN-Programme: db2expln und dynexpln.
- Der Anhang zu db2exfmt - dem Formatierungstool für EXPLAIN-Tabellen - beschreibt, wie mit diesem DB2-EXPLAIN-Tool EXPLAIN-Tabellendaten formatiert werden.

Teil 1. Datenbankkonzepte

Kapitel 1. Allgemeine Konzepte relationaler Datenbanken

Datenbankobjekte

Exemplare:

Ein *Exemplar* (zuweilen auch als *Datenbankmanager* bezeichnet) ist der DB2®-Code, der die Daten verwaltet. Er steuert, welche Operationen an den Daten ausgeführt werden können, und verwaltet die Systemressourcen, die ihm zugeordnet sind. Jedes Exemplar stellt eine vollständige Umgebung dar. Es enthält sämtliche Datenbankpartitionen, die für ein bestimmtes paralleles Datenbanksystem definiert sind. Ein Exemplar verfügt über eigene Datenbanken (auf die andere Exemplare keinen Zugriff haben), und alle zugehörigen Datenbankpartitionen benutzen gemeinsame Systemverzeichnisse. Es verfügt außerdem über ein von anderen Exemplaren auf derselben Maschine (System) getrenntes Sicherheitssystem.

Datenbanken:

Eine *relationale Datenbank* stellt Daten als Sammlung von Tabellen dar. Eine Tabelle besteht auf einer definierten Anzahl von Spalten und einer beliebigen Anzahl von Zeilen. Jede Datenbank enthält eine Gruppe von Systemkatalogtabellen, die die logische und physische Struktur der Daten beschreiben, eine Konfigurationsdatei, die die der Datenbank zugeordneten Parameterwerte enthält, sowie ein Wiederherstellungsprotokoll mit aktuellen und zu archivierenden Transaktionen.

Datenbankpartitionsgruppen:

Eine *Datenbankpartitionsgruppe* ist eine Gruppe aus einer oder mehreren Datenbankpartitionen. Wenn Sie Tabellen für die Datenbank erstellen wollen, erstellen Sie zunächst die Datenbankpartitionsgruppe, in der die Tabellenbereiche gespeichert werden. Anschließend erstellen Sie den Tabellenbereich, in dem die Tabellen gespeichert werden.

In früheren Versionen von DB2 UDB wurden *Datenbankpartitionsgruppen* als *Knotengruppen* bezeichnet.

Tabellenbereiche:

Eine Datenbank wird in Bestandteilen verwaltet, die als *Tabellenbereiche* bezeichnet werden. Ein Tabellenbereich ist ein Platz zum Speichern von Tabellen. Bei der Erstellung einer Tabelle können Sie sich entschließen, bestimmte Objekte wie Indizes und große Objekte (LOB-Daten) von den übrigen Tabellendaten getrennt zu speichern. Ein Tabellenbereich kann außerdem über eine oder mehrere physische Speichereinheiten verteilt angelegt werden. Das folgende Diagramm veranschaulicht etwas von der Flexibilität, die Sie durch die Verteilung von Daten auf Tabellenbereiche erhalten:

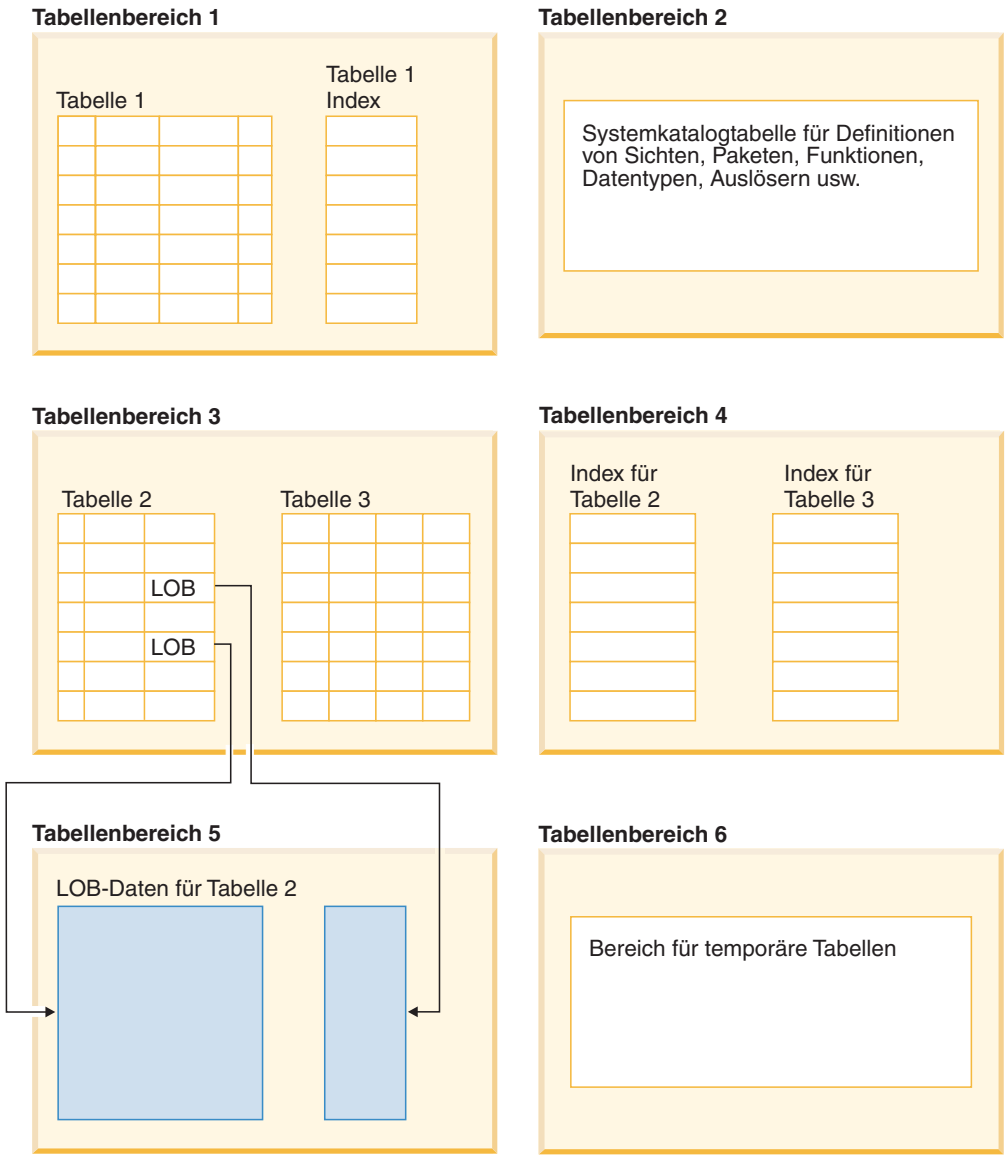


Abbildung 1. Flexibilität von Tabellenbereichen

Tabellenbereiche befinden sich in Datenbankpartitionsgruppen. Die Definitionen und Attribute von Tabellenbereichen werden im Systemkatalog der Datenbank aufgezeichnet.

Behälter werden Tabellenbereichen zugeordnet. Ein *Behälter* ist die Zuordnung eines physischen Speichers (z. B. eine Datei oder eine Einheit).

Ein Tabellenbereich kann entweder ein vom System verwalteter Bereich (SMS - System Managed Space) oder ein von der Datenbank verwalteter Bereich (DMS - Database Managed Space) sein. Bei einem SMS-Tabellenbereich ist jeder Behälter ein Verzeichnis im Dateibereich des Betriebssystems, wobei der Speicherbereich vom Dateimanager des Betriebssystems gesteuert wird. Bei einem DMS-Tabellenbereich ist jeder Behälter entweder eine im Voraus zugeordnete Datei fester Größe oder eine physische Einheit wie eine Platte, wobei der Datenbankmanager den Speicherbereich steuert.

Abb. 2 illustriert die Beziehung zwischen Tabellen, Tabellenbereichen und den beiden Typen von Speicherbereichen. Sie zeigt außerdem, dass Tabellen, Indizes und lange Daten (LOB-Daten) in Tabellenbereichen gespeichert werden.

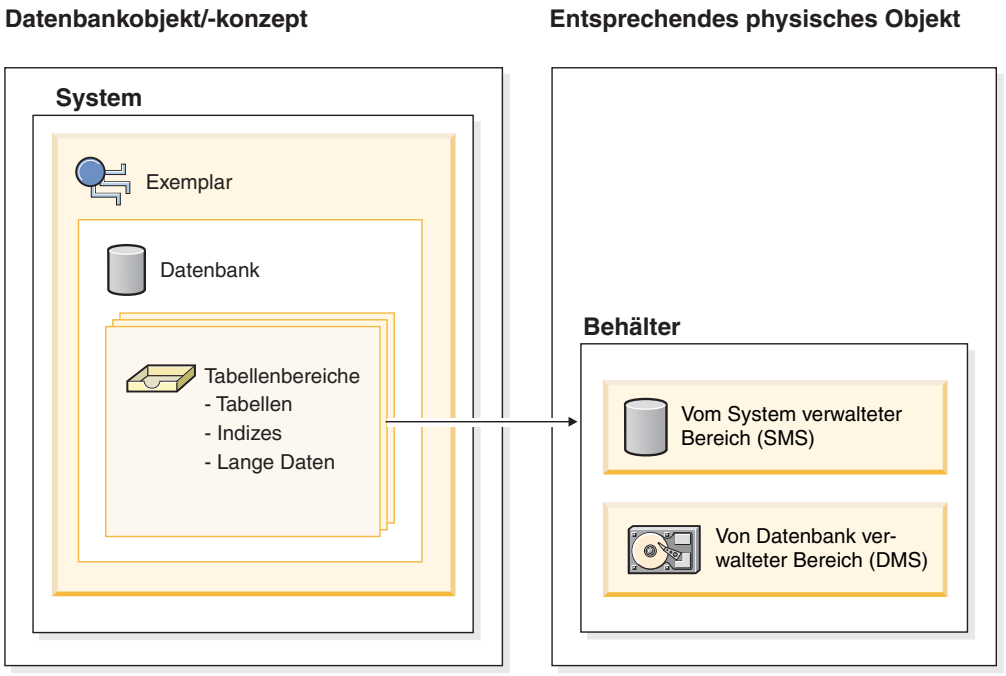


Abbildung 2. Tabellenbereiche und Behältertypen, die Daten enthalten

Abb. 3 auf Seite 6 zeigt die drei Typen von Tabellenbereichen: *REGULAR*, *TEMPORARY* und *LARGE*.

Tabellen, die Benutzerdaten enthalten, werden in regulären Tabellenbereichen gespeichert. Der Standardtabellenbereich für Benutzer besitzt den Namen USERSPACE1. Die Systemkatalogtabellen werden in einem regulären Tabellenbereich angelegt. Der Standardtabellenbereich für Systemkatalogtabellen heißt SYSCATSPACE.

Tabellen, die Spalten mit sehr langen Zeichenfolgen (Long) oder große Objekte (LOB) wie zum Beispiel Multimediaobjekte enthalten, werden in Tabellenbereichen für große Objektdaten (LARGE) oder in regulären Tabellenbereichen gespeichert. Die einfachen Spaltendaten für diese Spalten werden in einem regulären Tabellenbereich gespeichert, während die Langfelddaten oder großen Objekte im gleichen regulären Tabellenbereich oder in einem angegebenen großen Tabellenbereich (LARGE) gespeichert werden.

Indizes können in regulären oder großen Tabellenbereichen gespeichert werden.

Temporäre Tabellenbereiche werden als Systemtabellenbereiche oder Benutzertabellenbereiche klassifiziert. *Temporäre Systemtabellenbereiche* dienen zur Speicherung interner temporärer Daten, die für SQL-Operationen wie Sortieren, Reorganisieren von Tabellen, Erstellen von Indizes und Verknüpfen von Tabellen erforderlich sind. Obwohl eine beliebige Anzahl von temporären Systemtabellenbereichen erstellt werden kann, ist es zu empfehlen, nur einen solchen Tabellenbereich zu erstellen und dabei die Seitengröße zu verwenden, die von der Mehrheit der Tabellen verwendet wird.

Der Standardtabellenbereich für temporäre Systemdaten heißt TEMPSPACE1. *Temporäre Benutzertabellenbereiche* dienen zur Speicherung deklarierter globaler temporärer Tabellen, in denen temporäre Daten abgelegt werden. Temporäre Benutzertabellenbereiche werden *nicht* standardmäßig bei der Erstellung einer Datenbank erstellt.

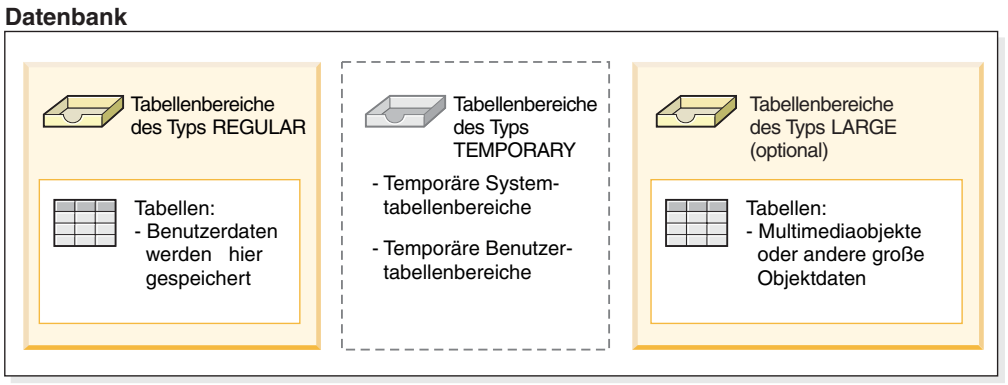


Abbildung 3. Typen von Tabellenbereichen

Tabellen:

Eine relationale Datenbank stellt Daten als Sammlung von Tabellen dar. Eine *Tabelle* besteht aus Daten, die logisch in Spalten und Zeilen angeordnet sind. Alle Datenbank- und Tabellendaten werden Tabellenbereichen zugeordnet. Die Daten in der Tabelle besitzen eine logische Beziehung, und zwischen Tabellen können Abhängigkeitsbedingungen definiert werden. Daten können nach mathematischen Prinzipien und mit Hilfe als *Relationen* bezeichneter Operationen in Sichten zusammengefasst und bearbeitet werden.

Der Zugriff auf Tabellendaten erfolgt mit Hilfe der Structured Query Language (SQL), einer standardisierten Sprache zur Definition und Bearbeitung von Daten in einer relationalen Datenbank. Eine *Abfrage* wird in Anwendungen oder von Benutzern verwendet, um Daten aus einer Datenbank abzurufen. In der Abfrage wird unter Verwendung von SQL eine Anweisung in folgendem Format erstellt:

```
SELECT <datenname> FROM <tabellenname>
```

Sichten:

Eine *Sicht* bietet eine effektive Methode zur Darstellung von Daten, ohne sie pflegen zu müssen. Eine Sicht ist keine wirkliche Tabelle und erfordert keine permanente Speicherung. Es wird eine „virtuelle Tabelle" erstellt und verwendet.

Eine Sicht kann alle oder einige der Spalten oder Zeilen der Tabelle enthalten, auf der sie basiert. Zum Beispiel können Sie eine Tabelle mit Abteilungsdaten (DEPARTMENT) und eine Tabelle mit Mitarbeiterdaten (EMPLOYEE) in einer Sicht verknüpfen, so dass Sie alle Mitarbeiter in einer bestimmten Abteilung auflisten können.

Abb. 4 auf Seite 7 zeigt die Beziehung zwischen Tabellen und Sichten.

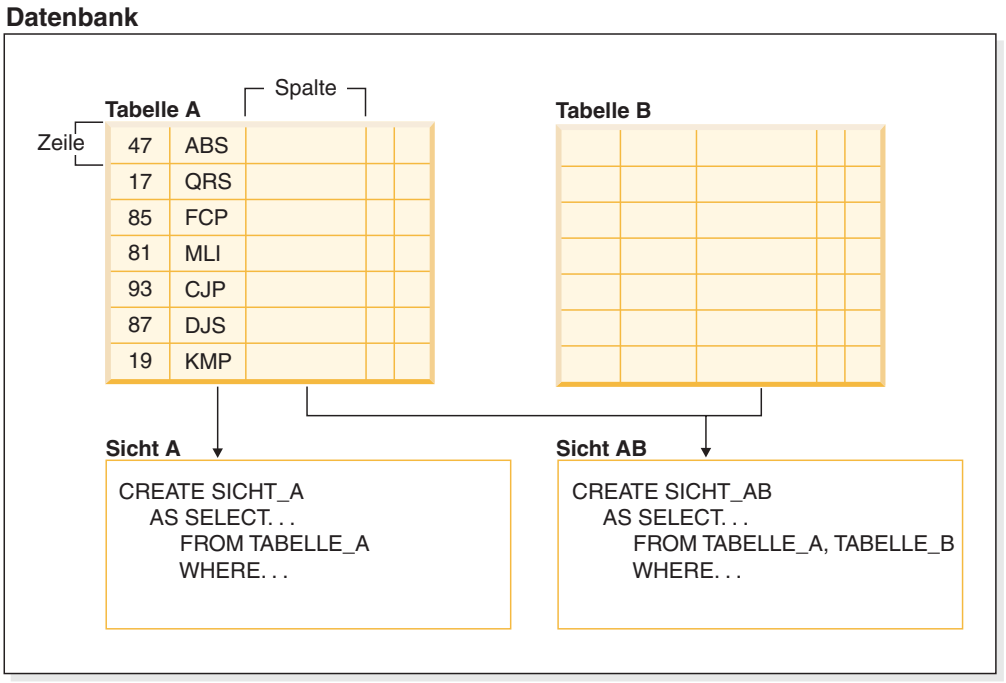


Abbildung 4. Beziehung zwischen Tabellen und Sichten

Indizes:

Ein *Index* ist ist eine Menge von Schlüsseln, die jeweils auf Zeilen in einer Tabelle zeigen. Zum Beispiel hat Tabelle A in Abb. 5 auf Seite 8 einen Index, der auf den Personalnummern in der Tabelle basiert. Dieser Schlüsselwert dient als Zeiger auf die Zeile in der Tabelle: die Personalnummer 19 zeigt auf Mitarbeiter KMP. Ein Index ermöglicht einen effizienteren Zugriff auf Zeilen einer Tabelle, indem er einen direkten Pfad zu den Daten mit Hilfe von Zeigern erstellt.

Das *SQL-Optimierungsprogramm* wählt automatisch den effizientesten Weg für den Zugriff auf Daten in Tabellen aus. Das Optimierungsprogramm bezieht Indizes bei der Ermittlung des schnellsten Pfads zu den Daten mit in Betracht.

Es können eindeutige Indizes erstellt werden, um die Eindeutigkeit des Indexschlüssels sicherzustellen. Ein *Indexschlüssel* ist eine Spalte oder eine geordnete Gruppe von Spalten, auf denen ein Index definiert wird. Mit Hilfe eines eindeutigen Index wird gewährleistet, dass der Wert jedes Indexschlüssels in der indizierten Spalte bzw. den indizierten Spalten eindeutig ist.

Abb. 5 auf Seite 8 illustriert die Beziehung zwischen einem Index und einer Tabelle.

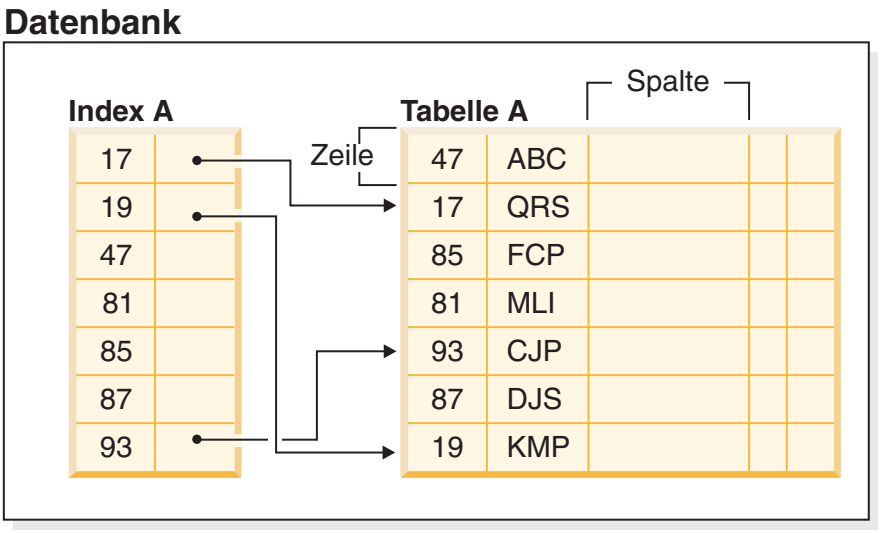


Abbildung 5. Beziehung zwischen einem Index und einer Tabelle

Abb. 6 veranschaulicht die Beziehungen zwischen einigen Datenbankobjekten. Sie zeigt außerdem, dass Tabellen, Indizes und lange Daten (LOB-Daten) in Tabellenbereichen gespeichert werden.

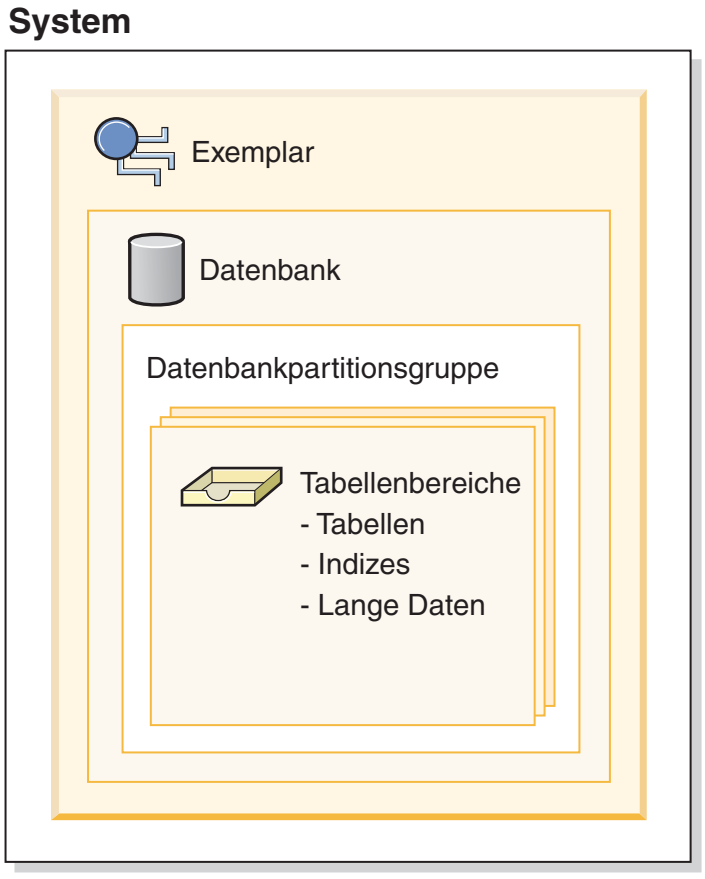


Abbildung 6. Beziehung zwischen ausgewählten Datenbankobjekten

Schemata:

Ein *Schema* ist eine Kennung, wie zum Beispiel eine Benutzer-ID, die bei der Gruppierung von Tabellen und anderen Datenbankobjekten hilfreich ist. Ein Schema kann einem Eigner zugeordnet sein, der den Zugriff auf die in ihm enthaltenen Daten und Objekte steuern kann.

Ein Schema ist außerdem ein Objekt in der Datenbank. Es kann automatisch erstellt werden, wenn das erste Objekt in einem Schema erstellt wird. Ein solches Objekt kann ein beliebiges Objekt sein, das durch einen Schemanamen qualifiziert werden kann, wie zum Beispiel eine Tabelle, ein Index, eine Sicht, ein Paket, ein einzigartiger Datentyp, eine Funktion oder ein Auslöser. Sie müssen über die Berechtigung IMPLICIT_SCHEMA verfügen, wenn das Schema automatisch erstellt werden soll. Sie können das Schema außerdem auch explizit erstellen.

Ein Schemaname wird als erster Teil eines zweiteiligen Objektnamens verwendet. Bei der Erstellung eines Objekts können Sie es einem bestimmten Schema zuordnen. Wenn Sie kein Schema angeben, wird das Objekt dem Standardschema zugeordnet, das in der Regel die Benutzer-ID der Person ist, die das Objekt erstellt hat. Der zweite Teil des Namens ist der Name des Objekts. Zum Beispiel könnte ein Benutzer namens Smith eine Tabelle des Namens SMITH.PAYROLL besitzen.

Systemkatalogtabellen:

Jede Datenbank enthält einen Satz von *Systemkatalogtabellen*, die die logische und physische Struktur der Daten beschreiben. DB2 UDB erstellt und pflegt einen umfangreichen Satz von Systemkatalogtabellen für jede einzelne Datenbank. Diese Tabellen enthalten Informationen über die Definitionen von Datenbankobjekten, wie zum Beispiel Benutzertabellen, Sichten und Indizes sowie Sicherheitsinformationen über die Berechtigungen, die Benutzer für diese Objekte besitzen. Sie werden bei der Erstellung der Datenbank erstellt und im Rahmen des normalen Datenbankbetriebs aktualisiert. Sie können nicht explizit erstellt oder gelöscht werden, jedoch kann ihr Inhalt mit Hilfe von Katalogsichten abgefragt und angezeigt werden.

Behälter:

Ein *Behälter* ist eine physische Speichereinheit. Er kann durch einen Verzeichnisnamen, einen Einheitennamen oder einen Dateinamen bezeichnet werden.

Ein Behälter ist einem Tabellenbereich zugeordnet. Ein einzelner Tabellenbereich kann sich über viele Behälter erstrecken, jedoch kann ein Behälter jeweils nur zu einem Tabellenbereich gehören.

Abb. 7 auf Seite 10 veranschaulicht die Beziehung zwischen Tabellen und einer Tabelle innerhalb einer Datenbank und den zugeordneten Behältern und Platten.

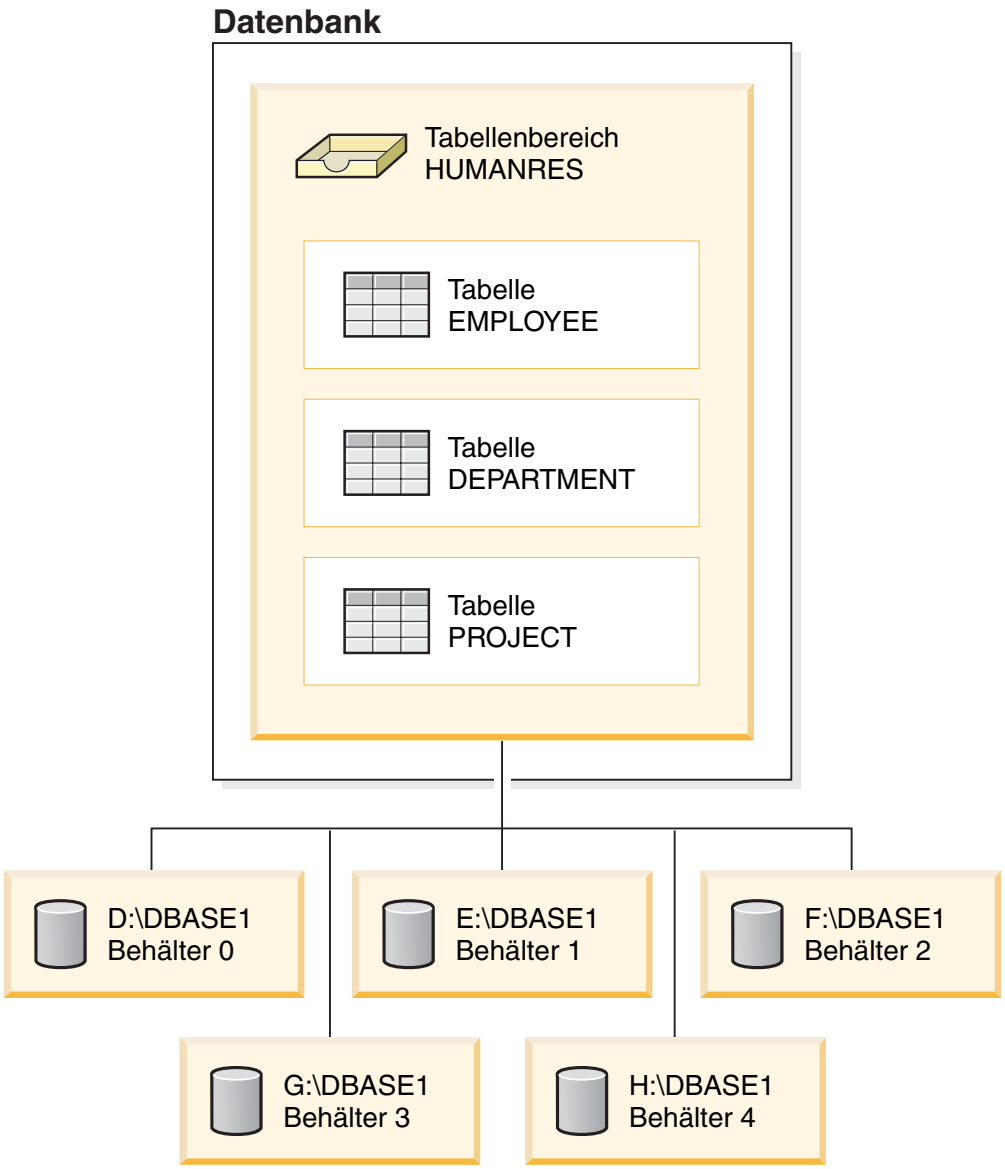


Abbildung 7. Beziehung zwischen einem Tabellenbereich und den zugehörigen Behältern

Die Tabellen EMPLOYEE, DEPARTMENT und PROJECT befinden sich im Tabellenbereich HUMANRES, der sich über die Behälter 0, 1, 2, 3 und 4 erstreckt. Dieses Beispiel zeigt jeden Behälter auf einer getrennten Platte.

Daten aller Tabellen werden in allen Behältern in einer Tabelle reihum verteilt gespeichert. Dies sorgt für eine gleichmäßige Verteilung der Daten auf die Behälter, die zu einem bestimmten Tabellenbereich gehören. Die Anzahl von Seiten, die der Datenbankmanager in einen Behälter schreibt, bevor er zu einem anderen Behälter wechselt, wird als *EXTENTSIZE* (Speicherbereichsgröße) bezeichnet.

Pufferpools:

Ein *Pufferpool* ist die Menge an Hauptspeicher, die beim Lesen von Daten vom Datenträger oder beim Ändern von Daten zum Zwischenspeichern von Tabellen- und Indexdatenseiten zugeordnet wird. Der Pufferpool dient zur Erhöhung der Systemleistung. Auf Daten im Hauptspeicher kann wesentlich schneller zugegriffen werden als auf Daten auf einem Datenträger. Daher ist die Leistung umso höher, je

weniger oft der Datenbankmanager Daten von einem Datenträger lesen bzw. auf ihn schreiben (E/A) muss. (Sie können mehr als einen Pufferpool erstellen, wenngleich für die meisten Fälle nur ein Pufferpool erforderlich ist.)

Die Konfiguration des Pufferpools ist der wichtigste Einzelbereich der Optimierung, weil hier die Verzögerung durch eine langsame Ein-/Ausgabe (E/A) verringert werden kann.

Abb. 8 veranschaulicht die Beziehung zwischen einem Pufferpool und Behältern.

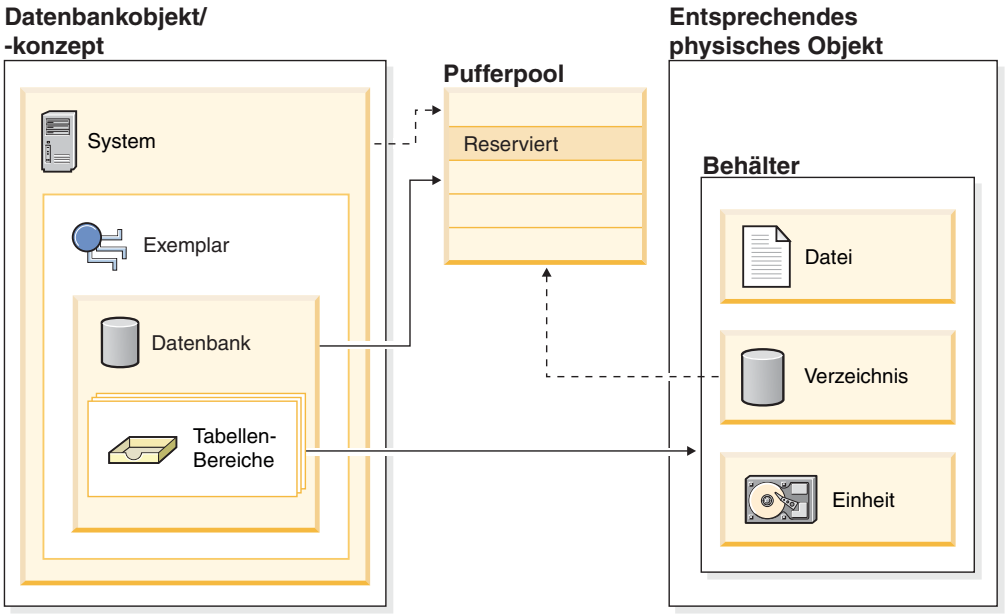


Abbildung 8. Beziehung zwischen dem Pufferpool und Behältern

Zugehörige Konzepte:

- „Indexes" in *SQL Reference, Volume 1*
- „Tables" in *SQL Reference, Volume 1*
- „Relational databases" in *SQL Reference, Volume 1*
- „Schemas" in *SQL Reference, Volume 1*
- „Views" in *SQL Reference, Volume 1*
- „Table spaces and other storage structures" in *SQL Reference, Volume 1*

Konfigurationsparameter

Bei der Erstellung eines DB2™-Exemplars oder einer DB2-Datenbank wird eine entsprechende Konfigurationsdatei mit den Standardparameterwerten erstellt. Diese Parameterwerte können zur Verbesserung der Leistung und anderer Merkmale des Exemplars oder der Datenbank geändert werden.

Konfigurationsdateien enthalten Parameter, die Werte definieren, wie zum Beispiel die den DB2 UDB-Produkten 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 und sind von keiner Datenbank, die zu diesem Exemplar gehört, abhängig. 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.

Konfigurationsparameter des Datenbankmanagers werden in einer Datei mit dem Namen `db2systm` gespeichert. Diese Datei wird erstellt, wenn das Exemplar des Datenbankmanagers erstellt wird. In UNIX-basierten Umgebungen befindet sich diese Datei im Unterverzeichnis `sqllib` für das Exemplar des Datenbankmanagers. Unter Windows ist die Standardposition dieser Datei das Exemplarunterverzeichnis des Verzeichnisses `sqllib`. Wenn die Variable DB2INSTPROF definiert ist, befindet sich die Datei im Unterverzeichnis `instance` des Verzeichnisses, das durch die Variable DB2INSTPROF angegeben wird.

In einer Umgebung mit partitionierten Datenbanken befindet sich diese Datei in einem gemeinsamen Dateisystem, so dass alle Datenbankpartitionsserver Zugriff auf dieselbe Datei haben. Die Konfiguration des Datenbankmanagers ist auf allen Datenbankpartitionsservern identisch.

Die meisten Parameter haben entweder Einfluss darauf, wie viel Systemressourcen einem einzelnen Datenbankmanagerexemplar zugeordnet werden, oder sie konfigurieren die Einrichtung des Datenbankmanagers und der verschiedenen Kommunikationssubsysteme nach umgebungsspezifischen Überlegungen. 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 auf den einzelnen Datenbankpartitionen übereinstimmen oder variieren, jedoch wird empfohlen, dass die Parameterwerte für die Datenbankkonfiguration in allen Partitionen übereinstimmen sollten.

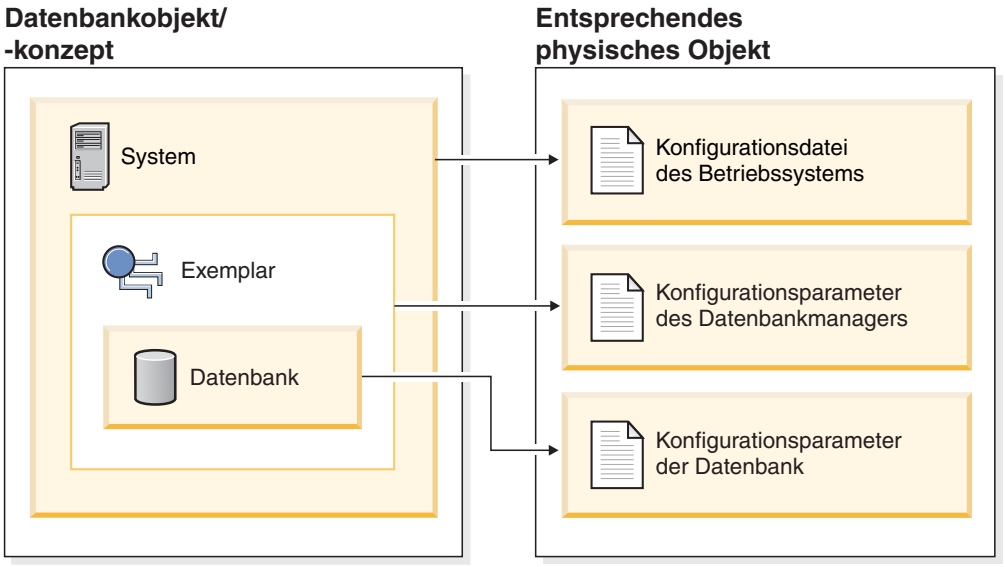


Abbildung 9. Beziehung zwischen Datenbankobjekten und Konfigurationsdateien

Zugehörige Konzepte:

- „Konfigurieren der Parameteroptimierung" in *Systemverwaltung: Optimierung*

Zugehörige Tasks:

- „Konfigurieren von DB2 mit Konfigurationsparametern" in *Systemverwaltung: Optimierung*

Geschäftsregeln für Daten

In jeder Geschäftsumgebung müssen Daten häufig bestimmten Rahmenbedingungen oder Regeln genügen. Zum Beispiel muss eine Personalnummer eindeutig sein. DB2® Universal Database (DB2 UDB) stellt *Integritätsbedingungen* (engl. Constraints) bereit, die zur Implementierung solcher Regeln definiert werden können.

DB2 UDB stellt die folgenden Arten von Integritätsbedingungen zur Verfügung:

- Integritätsbedingung NOT NULL
- Eindeutige Integritätsbedingung
- Integritätsbedingung über Primärschlüssel
- Integritätsbedingung über Fremdschlüssel
- Prüfung auf Integritätsbedingung
- Informative Integritätsbedingung

Integritätsbedingung NOT NULL
Die Integritätsbedingung NOT NULL verhindert, dass Nullwerte in eine Spalte eingegeben werden.

Eindeutige Integritätsbedingung

Eindeutige Integritätsbedingungen stellen sicher, dass die Werte in einer Gruppe von Spalten eindeutig sind und für alle Zeilen in der Tabelle nicht null sind. Zum Beispiel könnte eine typische eindeutige Integritätsbedingung in einer Tabelle DEPARTMENT mit Daten über Abteilungen darin bestehen, dass die Abteilungsnummer (DEPTNO) eindeutig und nicht null ist.

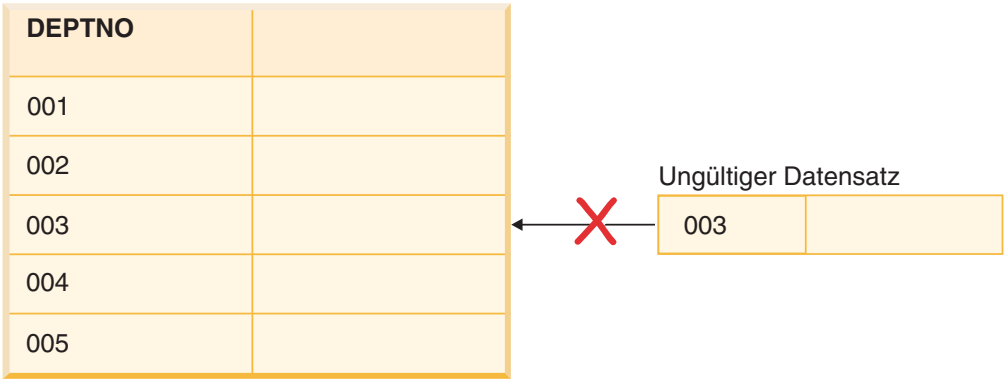


Abbildung 10. Eindeutige Integritätsbedingungen verhindern Datenduplikate

Der Datenbankmanager beachtet die Integritätsbedingung bei Operationen zum Einfügen und Aktualisieren von Daten, um die Datenintegrität zu gewährleisten.

Integritätsbedingung über Primärschlüssel

Jede Tabelle kann einen und nur einen Primärschlüssel besitzen. Ein Primärschlüssel ist eine Spalte oder eine Kombination von Spalten, die die gleichen Merkmale wie eine eindeutige Integritätsbedingung besitzen. Ein Primärschlüssel und Integritätsbedingungen über Fremdschlüssel dienen zur Definition von Beziehungen zwischen Tabellen.

Da ein Primärschlüssel zur Angabe einer Zeile in einer Tabelle verwendet wird, sollte er eindeutig sein und möglichst wenigen Hinzufüge- oder Löschoperationen unterliegen. Eine Tabelle kann nicht mehr als einen Primärschlüssel, jedoch mehrere eindeutige Schlüssel besitzen. Primärschlüssel sind wahlfrei und können definiert werden, wenn eine Tabelle erstellt oder geändert wird. Sie besitzen zudem den weiteren Vorteil, dass sie für eine Reihenfolge der Daten sorgen, wenn Daten exportiert oder reorganisiert werden.

In den folgenden Tabellen sind DEPTNO und EMPNO die Primärschlüssel der Tabellen DEPARTMENT und EMPLOYEE.

Tabelle 1. Tabelle DEPARTMENT

DEPTNO (Primärschlüssel)	DEPTNAME	MGRNO
A00	Spiffy Computer Service Division	000010
B01	Planning	000020
C01	Information Center	000030
D11	Manufacturing Systems	000060

Tabelle 2. Tabelle EMPLOYEE

EMPNO (Primär-schlüssel)	FIRSTNAME	LASTNAME	WORKDEPT (Fremd-schlüssel)	PHONENO
000010	Christine	Haas	A00	3978
000030	Sally	Kwan	C01	4738
000060	Irving	Stern	D11	6423
000120	Sean	O'Connell	A00	2167
000140	Heather	Nicholls	C01	1793
000170	Masatoshi	Yoshimura	D11	2890

Integritätsbedingung über Fremdschlüssel

Integritätsbedingungen über Fremdschlüssel (die auch als referenzielle Integritätsbedingungen bezeichnet werden) geben Ihnen die Möglichkeit, erforderliche Beziehungen zwischen und innerhalb von Tabellen zu definieren.

Zum Beispiel könnte eine typische Integritätsbedingung über Fremdschlüssel festlegen, dass jeder Mitarbeiter in der Tabelle EMPLOYEE ein Mitglied einer bestehenden, in der Tabelle DEPARTMENT definierten Abteilung sein muss.

Zur Herstellung dieser Beziehung würden Sie die Abteilungsnummer (WORKDEPT) der Tabelle EMPLOYEE als Fremdschlüssel und die Abteilungsnummer (DEPTNO) der Tabelle DEPARTMENT als Primärschlüssel definieren.

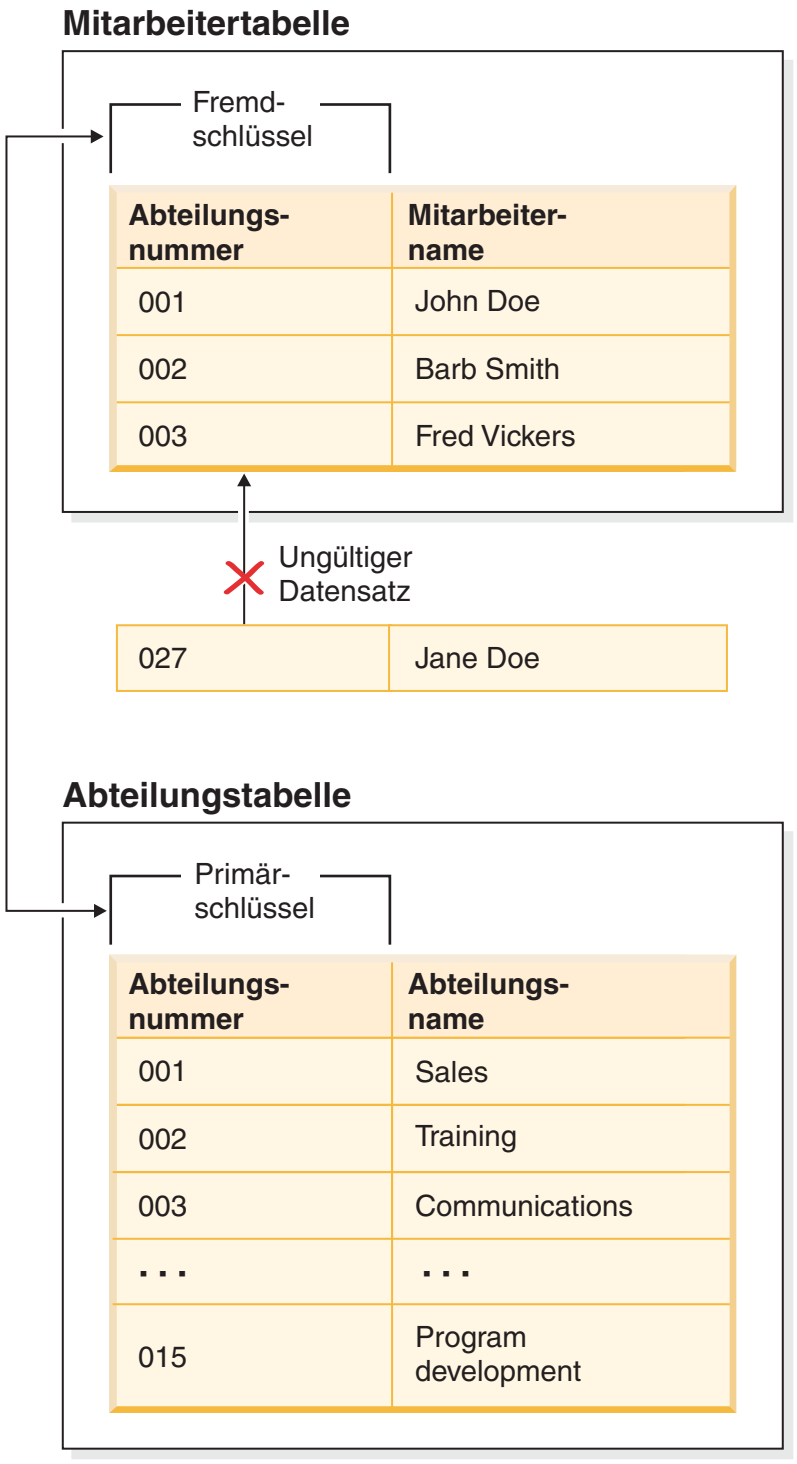


Abbildung 11. Integritätsbedingungen über Fremd- und Primärschlüssel

Prüfung auf Integritätsbedingung

Eine Prüfung auf Integritätsbedingung ist eine Datenbankregel, mit der die zulässigen Werte in einer oder mehreren Spalten jeder Zeile einer Tabelle angegeben werden.

Zum Beispiel können Sie einer Tabelle EMPLOYEE definieren, dass die Spalte für die Jobbezeichnung nur die Werte ″Sales″, ″Manager″ oder ″Clerk″ enthalten kann. Unter dieser Integritätsbedingung ist jeder

Datensatz mit einem anderen Wert in der Spalte für die Jobbezeichnung ungültig und würde zurückgewiesen, um die Regeln über den Typ der zulässigen Daten in der Tabelle durchzusetzen.

Informative Integritätsbedingung
Eine informative Integritätsbedingung ist eine Regel, die vom SQL-Compiler verwendet werden kann, jedoch vom Datenbankmanager nicht umgesetzt wird. Der Zweck der Integritätsbedingung liegt nicht darin, eine zusätzliche Datenprüfung durch den Datenbankmanager zu bewirken, sondern darin, die Abfrageleistung zu verbessern.

Informative Integritätsbedingungen werden mit Hilfe der Anweisungen CREATE TABLE oder ALTER TABLE definiert. Sie fügen eine referenzielle Integritätsbedingungen oder Prüfungen auf Integritätsbedingungen hinzu und ordnen diesen dann Integritätsbedingungsattribute zu, indem Sie angeben, ob der Datenbankmanager die Integritätsbedingung umsetzen soll oder nicht und ob die Integritätsbedingung zur Abfrageoptimierung verwendet werden soll oder nicht.

In einer Datenbank können außerdem *Auslöser* genutzt werden. Auslöser sind komplizierter und potenziell leistungsfähiger als Integritätsbedingungen. Sie definieren eine Reihe von Aktionen, die in Verbindung mit einer bzw. ausgelöst durch eine Klausel INSERT, UPDATE oder DELETE an einer angegebenen Basistabelle ausgeführt wird. Auslöser können zur Unterstützung allgemeiner Formen der Datenintegrität und Geschäftsregeln verwendet werden. Zum Beispiel kann ein Auslöser die Kreditgrenze eines Kunden überprüfen, bevor ein Auftrag akzeptiert wird, oder er kann in einer Bankanwendung eingesetzt werden, um einen Alert auszulösen, wenn eine Geldentnahme aus einem Konto nicht dem gewöhnlichen Geldentnahmemuster eines Kunden entspricht.

Zugehörige Konzepte:

- „Integritätsbedingungen" auf Seite 69
- „Auslöser" auf Seite 74

Entwickeln einer Sicherungs- und Wiederherstellungsstrategie

Eine Datenbank kann aufgrund von Hardware- oder Softwarefehlern (oder beidem) unbrauchbar werden. Irgendwann treten möglicherweise Speicherprobleme, Stromausfälle und Anwendungsfehler auf, und jedes Fehlerszenario erfordert eine andere Wiederherstellungsaktion. Schützen Sie Ihre Daten vor Verlust, indem Sie eine gut erprobte Wiederherstellungsstrategie bereithalten. Bei der Entwicklung Ihrer Wiederherstellungsstrategie sollten Sie unter anderem folgende Fragen berücksichtigen:

- Soll die Datenbank wiederherstellbar sein?
- Wie viel Zeit steht für die Wiederherstellung der Datenbank zur Verfügung?
- In welchen Abständen werden Sicherungsoperationen durchgeführt?
- Wie viel Speicher kann für Sicherungskopien und Archivprotokolldateien zugeordnet werden?
- Sind Sicherungen auf Tabellenbereichsebene ausreichend oder muss die gesamte Datenbank gesichert werden?
- Sollte ein Bereitschaftssystem manuell oder über HADR (High Availability Disaster Recovery) konfiguriert werden?

Eine Strategie zur Wiederherstellung der Datenbank sollte sicherstellen, dass alle Informationen verfügbar sind, wenn sie für eine Wiederherstellung der Datenbank benötigt werden. Sie sollte einen Zeitplan für regelmäßige Sicherungen enthalten. Im Fall von partitionierten Datenbanksystemen sollten auch nach einer Skalierung des Systems, d. h. nach dem Hinzufügen eines Datenbankpartitionsservers oder -knotens, Sicherungen durchgeführt werden. Ihre Gesamtstrategie sollte auch Prozeduren zum Wiederherstellen von Befehlsprozeduren, Anwendungen, benutzerdefinierten Funktionen, Code gespeicherter Prozeduren in Betriebssystembibliotheken sowie von Ladekopien enthalten.

In den folgenden Abschnitten werden verschiedene Wiederherstellungsmethoden behandelt. Sie können bestimmen, welche Wiederherstellungsmethode für Ihre Geschäftsumgebung am besten geeignet ist.

Das Konzept einer *Datenbanksicherung* ist mit jeder anderen Datensicherung vergleichbar: Sie erstellen eine Kopie der Daten und speichern die Kopie anschließend auf einem anderen Datenträger für den Fall eines Ausfalls oder einer Beschädigung der Originaldaten. Den einfachsten Fall einer Sicherung bildet das Verfahren, bei dem zunächst die Datenbank gestoppt wird, um sicherzustellen, dass keine weiteren Transaktionen auftreten, und anschließend ein Sicherungsimage der Daten erstellt wird. Sie können die Datenbank dann erneut erstellen, falls sie beschädigt wird.

Das erneute Erstellen der Datenbank wird als *Wiederherstellung* bezeichnet. Eine *Versionswiederherstellung* ist die Wiederherstellung einer früheren Version der Datenbank mit Hilfe eines Images der Datenbank, das im Rahmen einer Sicherungsoperation erstellt wurde. Eine *aktualisierende Wiederherstellung* ist die erneute Anwendung von in den Datenbankprotokolldateien aufgezeichneten Transaktionen nach der Wiederherstellung eines Sicherungsimages einer Datenbank oder eines Tabellenbereichs.

Eine *Wiederherstellung nach einem Systemabsturz* ist die automatische Wiederherstellung der Datenbank, wenn ein Fehler auftritt, bevor alle Änderungen einer oder mehrerer Arbeitseinheiten (Transaktionen) beendet und festgeschrieben wurden. Dies geschieht durch Rückgängigmachen der unvollständigen Transaktionen und Beenden der festgeschriebenen Aktionen, die sich noch im Hauptspeicher befanden, als der Systemabsturz auftrat.

Die Protokolldateien für die Wiederherstellung und die Datei des Wiederherstellungsprotokolls werden automatisch erstellt, wenn eine Datenbank erstellt wird (Abb. 12 auf Seite 19). Diese Protokolldateien sind wichtig, falls Sie verlorene oder beschädigte Daten wiederherstellen müssen.

Jede Datenbank enthält *Wiederherstellungsprotokolle*, die zur Datenwiederherstellung nach Anwendungs- oder Systemfehlern verwendet werden. In Kombination mit den Datenbanksicherungen dienen sie zur Wiederherstellung der Konsistenz der Datenbank bis exakt zu dem Zeitpunkt, an dem der Fehler auftrat.

Die *Datei des Wiederherstellungsprotokolls* enthält eine Zusammenfassung der Sicherungsinformationen, die zur Bestimmung der Wiederherstellungsoptionen verwendet werden können, wenn die Datenbank insgesamt oder teilweise bis zu einem bestimmten Zeitpunkt wiederhergestellt werden muss. Sie dient zur Protokollierung wiederherstellungsrelevanter Ereignisse wie z. B. Sicherungs- und Wiederherstellungsoperationen. Diese Datei befindet sich im Datenbankverzeichnis.

Die *Protokolldatei der Tabellenbereichsänderungen*, die sich ebenfalls im Datenbankverzeichnis befindet, enthält Informationen, anhand deren bestimmt werden kann, welche Protokolldateien zur Wiederherstellung eines bestimmten Tabellenbereichs erforderlich sind.

Die Datei des Wiederherstellungsprotokolls oder die Protokolldatei der Tabellenbereichsänderungen können nicht direkt geändert werden. Mit dem Befehl PRUNE HISTORY können Sie jedoch Einträge aus den Dateien löschen. Sie haben auch die Möglichkeit, mit dem Datenbankkonfigurationsparameter *rec_his_retentn* die Anzahl Tage anzugeben, die diese Protokolldateien beibehalten werden sollen.

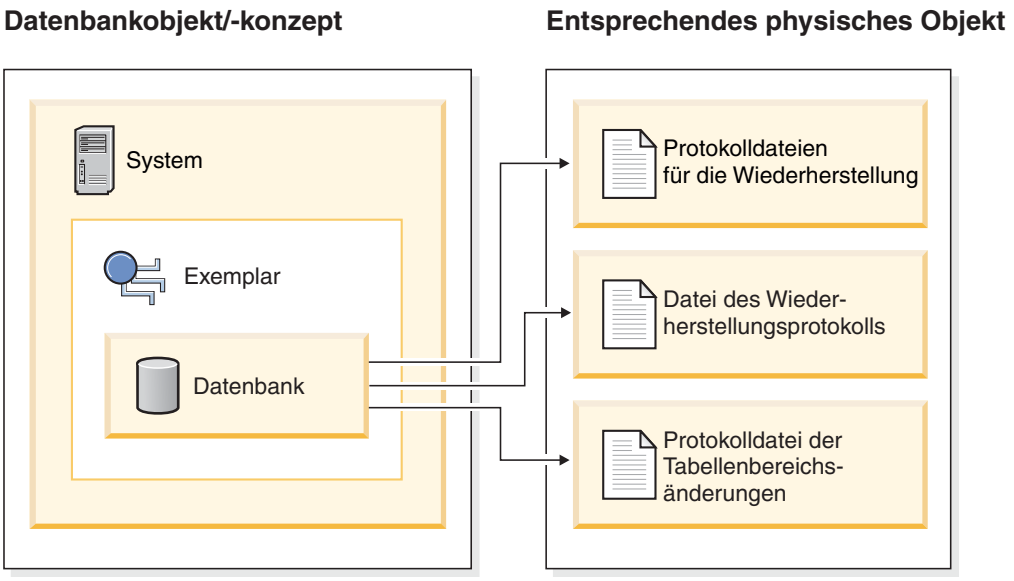


Abbildung 12. Dateien zur Datenbankwiederherstellung

Daten, die sich problemlos erneut erstellen lassen, können in einer nicht wiederherstellbaren Datenbank gespeichert werden. Dazu gehören Daten aus einer externen Quelle, die für Anwendungen mit Lesezugriff verwendet werden, und Tabellen, die nicht oft aktualisiert werden und für die der geringe Protokollierungsaufwand nicht die zusätzliche Komplexität der Verwaltung von Protokolldateien sowie die aktualisierende Wiederherstellung nach einer Wiederherstellungsoperation rechtfertigt. Bei *nicht wiederherstellbaren Datenbanken* sind die Datenbankkonfigurationsparameter *logarchmeth1* und *logarchmeth2* auf den Wert „OFF" gesetzt. Dies bedeutet, dass lediglich die für die Wiederherstellung nach einem Systemabsturz erforderlichen Protokolle beibehalten werden. Diese Protokolle werden als *aktive Protokolldateien* bezeichnet und enthalten aktuelle Transaktionsdaten. Die Versionswiederherstellung mit Hilfe von *Offlinesicherungen* ist das primäre Mittel zur Wiederherstellung einer nicht wiederherstellbaren Datenbank. (Eine Offlinesicherung bedeutet, dass während der Sicherungsoperation keine andere Anwendung die Datenbank verwenden kann.) Eine solche Datenbank kann nur offline wiederhergestellt werden. Sie wird in dem Status wiederhergestellt, den sie hatte, als das Sicherungsimage erstellt wurde. Eine aktualisierende Wiederherstellung wird nicht unterstützt.

Daten, die sich *nicht* einfach erneut erstellen lassen, sollten in einer wiederherstellbaren Datenbank gespeichert werden. Hierzu gehören Daten, deren Quelle nach dem Laden der Daten zerstört wird, Daten, die manuell in Tabellen eingegeben werden, sowie Daten, die nach dem Laden in die Datenbank von Anwendungsprogrammen oder Benutzern geändert werden. Bei *wiederherstellbaren Datenbanken*

sind die Datenbankkonfigurationsparameter *logarchmeth1* oder *logarchmeth2* auf einen anderen Wert als „OFF" gesetzt. Aktive Protokolldateien sind weiterhin für die Datenbankwiederherstellung nach einem Systemabsturz verfügbar. Es stehen jedoch auch die *Archivprotokolldateien* zur Verfügung, die festgeschriebene Transaktionsdaten enthalten. Eine solche Datenbank kann nur offline wiederhergestellt werden. Sie wird in dem Status wiederhergestellt, den sie hatte, als das Sicherungsimage erstellt wurde. Mit Hilfe der aktualisierenden Wiederherstellung (Rollforward Recovery) können Sie jedoch die Datenbank *aktualisierend wiederherstellen* (also über den Zeitpunkt hinaus, an dem das Sicherungsimage erstellt wurde), indem Sie die aktiven und archivierten Protokolle entweder bis zu einem bestimmten Zeitpunkt oder bis zum Ende der aktiven Protokolldateien anwenden.

Sicherungsoperationen für wiederherstellbare Datenbanken können entweder offline oder *online* durchgeführt werden (online heißt, dass andere Anwendungen während der Sicherungsoperation eine Verbindung zur Datenbank herstellen können). Onlineoperationen zur Tabellenbereichswiederherstellung und zur aktualisierenden Wiederherstellung werden nur unterstützt, wenn die Datenbank als wiederherstellbar definiert ist. Ist die Datenbank nicht wiederherstellbar, müssen Operationen zur Datenbankwiederherstellung und aktualisierenden Wiederherstellung offline ausgeführt werden. Während einer Onlinesicherung stellt die aktualisierende Wiederherstellung sicher, dass *alle* Tabellenänderungen erfasst und erneut angewendet werden, wenn diese Sicherungskopie wiederhergestellt wird.

Bei einer wiederherstellbaren Datenbank können Sie anstelle der gesamten Datenbank auch einzelne Tabellenbereiche sichern, wiederherstellen und aktualisierend wiederherstellen. Wenn Sie einen Tabellenbereich online sichern, kann er weiterhin verwendet werden, und gleichzeitig durchgeführte Änderungen werden in den Protokollen aufgezeichnet. Wenn Sie einen Tabellenbereich online wiederherstellen oder online aktualisierend wiederherstellen, kann der Tabellenbereich selbst nicht mehr verwendet werden, bis die Operation beendet ist, Benutzer können jedoch weiterhin auf Tabellen in anderen Tabellenbereichen zugreifen.

Automatisierte Sicherungsoperationen

Da die Bestimmung, ob und wann Verwaltungsaktivitäten wie zum Beispiel Sicherungsoperationen auszuführen sind, zeitaufwendig sein kann, haben Sie die Möglichkeit, diese Aufgabe durch den Assistenten *Automatische Verwaltung konfigurieren* erledigen zu lassen. Bei der automatischen Verwaltung definieren Sie Ihre Verwaltungszielsetzungen, einschließlich der möglichen Ausführungszeiten für die automatische Verwaltung. DB2 bestimmt anhand dieser Verwaltungszielsetzungen, ob die Verwaltungsaktivitäten ausgeführt werden müssen, und führt anschließend während des nächsten verfügbaren Verwaltungsfensters (ein benutzerdefinierter Zeitraum zur Ausführung automatischer Verwaltungsaktivitäten) nur die erforderlichen Verwaltungsaktivitäten aus.

Anmerkung: Nach der Konfiguration der automatischen Verwaltung haben Sie weiterhin die Möglichkeit, manuelle Sicherungsoperationen ausführen. DB2 führt automatische Sicherungsoperationen nur dann aus, wenn sie erforderlich sind.

Zugehörige Konzepte:

- „Wiederherstellung nach einem Systemabsturz" in *Datenwiederherstellung und hohe Verfügbarkeit Handbuch und Referenz*
- „Versionswiederherstellung" in *Datenwiederherstellung und hohe Verfügbarkeit Handbuch und Referenz*

- „Aktualisierende Wiederherstellung" in *Datenwiederherstellung und hohe Verfügbarkeit Handbuch und Referenz*
- „HADR (High Availability Disaster Recovery) - Übersicht" in *Datenwiederherstellung und hohe Verfügbarkeit Handbuch und Referenz*
- „Data Links server file backups" in *DB2 Data Links Manager Administration Guide and Reference*
- „Failure and recovery overview" in *DB2 Data Links Manager Administration Guide and Reference*

Zugehörige Referenzen:

- „rec_his_retentn - Aufbewahrungszeitraum für Wiederherstellungsprotokoll" in *Systemverwaltung: Optimierung*
- „logarchmeth1 - Primäre Protokollarchivierungsmethode (Konfigurationsparameter)" in *Systemverwaltung: Optimierung*
- „DB2 Data Links Manager system setup and backup recommendations" in *DB2 Data Links Manager Administration Guide and Reference*

Automatische Verwaltung

DB2® Universal Database (DB2 UDB) stellt Funktionen zur automatischen Verwaltung zur Verfügung, um bei Bedarf Datenbanksicherungen, Statistikaktualisierungen und Reorganisationen von Tabellen und Indizes auszuführen.

Die automatische Datenbanksicherung stellt eine Lösung für Benutzer bereit, die ihnen bei der Sicherstellung hilft, dass ihre Datenbank sowohl ordnungsgemäß als auch regelmäßig gesichert wird, ohne sich um den Zeitpunkt der Sicherung kümmern oder den Befehl BACKUP kennen zu müssen.

Die automatische Statistikerfassung versucht, die Leistung der Datenbank durch die Pflege aktueller Tabellenstatistiken zu verbessern. Dies dient dazu, dem Optimierungsprogramm die Möglichkeit zu geben, einen Zugriffsplan auf der Grundlage exakter Statistikdaten auszuwählen.

Die automatische Statistikprofilerstellung gibt Empfehlungen, wann und wie Tabellenstatistiken zu erfassen sind, indem sie veraltete, fehlende und falsch spezifizierte Statistikdaten erkennt und Statistikprofile auf der Basis des Abfragefeedbacks generiert.

Die automatische Reorganisation verwaltet eine Offlinereorganisation von Tabellen und Indizes, ohne dass sich Benutzer darum zu kümmern brauchen, wann und wie ihre Daten zu reorganisieren sind.

Die Aktivierung dieser automatischen Verwaltungsfunktionen wird über die Datenbankkonfigurationsparameter zur automatischen Verwaltung gesteuert. Bei ihnen handelt es sich um eine hierarchische Gruppe von Schaltern, die eine einfache und flexible Verwaltung der Aktivierung dieser Funktionen ermöglichen.

Automatische Datenbanksicherung:

Eine Datenbank kann durch eine Vielzahl möglicher Hard- und Softwarefehler unbrauchbar werden. Die automatische Datenbanksicherung vereinfacht die Verwaltung von Datenbanksicherungsaufgaben für den Datenbankadministrator (DBA), indem sie fortlaufend sicherstellt, dass bei Bedarf eine neue Gesamt-

sicherung der Datenbank durchgeführt wird. Sie stellt auf der Grundlage eines oder mehrerer der folgenden Kriterien fest, ob eine Sicherungsoperation ausgeführt werden muss:

- Sie haben nie eine Gesamtsicherung der Datenbank durchgeführt.
- Die abgelaufene Zeit seit der letzten Gesamtsicherung ist länger als die angegebene Anzahl von Stunden.
- Der Speicherplatz für Transaktionsprotokolle seit der letzten Sicherung ist größer als die angegebene Anzahl von 4-KB-Seiten (nur im Protokollarchivierungsmodus).

Schützen Sie Ihre Daten, indem Sie eine Strategie zur Wiederherstellung nach einem Katastrophenfall für Ihr System planen und implementieren. Wenn dies Ihren Anforderungen entgegenkommt, können Sie die Funktion zur automatischen Datenbanksicherung in Ihre Sicherungs- und Wiederherstellungsstrategie integrieren.

Wenn die Datenbank für die aktualisierende Wiederherstellung (Protokollarchivierung) eingerichtet ist, kann die automatische Datenbanksicherung entweder für die Onlinesicherung oder für die Offlinesicherung aktiviert werden. Ansonsten ist nur die Offlinesicherung verfügbar. Die automatische Datenbanksicherung unterstützt Plattenspeicher, Bandspeicher, Tivoli® Storage Manager (TSM) und DLL-Datenträgertypen von Fremdanbietern.

Über den Assistenten „Automatische Verwaltung konfigurieren" in der Steuerzentrale oder der Diagnosezentrale können Sie folgende Merkmale konfigurieren:

- Die erforderliche Zeit oder Anzahl von Protokollseiten zwischen den Sicherungen
- Die Sicherungsdatenträger
- Online- oder Offlinesicherung

Wenn eine Sicherung auf Platte ausgewählt wird, löscht die automatische Sicherungsfunktion die Sicherungsimages regelmäßig aus dem im Assistenten „Automatische Verwaltung konfigurieren" angegebenen Verzeichnis. Daher steht zu einem Zeitpunkt nur das letzte Sicherungsimage garantiert zur Verfügung. Es wird empfohlen, dieses Verzeichnis ausschließlich für die automatische Sicherungsfunktion zu reservieren und nicht zur Speicherung anderer Sicherungsimages zu verwenden.

Die Funktion zur automatischen Datenbanksicherung kann mit Hilfe der Datenbankkonfigurationsparameter **auto_db_backup** und **auto_maint** aktiviert bzw. inaktiviert werden. In einer Umgebung mit mehreren Datenbankpartitionen wird die automatische Datenbanksicherung in jeder Partition ausgeführt, wenn die Datenbankkonfigurationsparameter für die jeweilige Partition aktiviert sind.

Automatische Statistikerfassung:

Wenn der SQL-Compiler SQL-Abfragepläne optimiert, werden die getroffenen Entscheidungen erheblich durch statistische Informationen zur Größe der Datenbanktabellen und Indizes beeinflusst. Das Optimierungsprogramm nutzt zudem Informationen zur Verteilung von Daten in bestimmten Spalten von Tabellen und Indizes, wenn diese Spalten zur Auswahl von Zeilen bzw. zur Verknüpfung von Tabellen dienen. Das Optimierungsprogramm verwendet diese Informationen zur Abschätzung des Aufwands alternativer Zugriffspläne für eine Abfrage. Wenn Tabellenzeilen in großen Mengen hinzugefügt oder entfernt werden oder wenn

Daten in Spalten, für die Sie Statistiken erfassen, aktualisiert werden, muss das Dienstprogramm RUNSTATS erneut ausgeführt werden, um die Statistiken zu aktualisieren.

Die automatische Statistikerfassung arbeitet, indem sie die minimale Gruppe von Statistiken bestimmt, die eine optimale Leistungsverbesserung ermöglicht. Die Entscheidung, Statistiken zu erfassen bzw. zu aktualisieren, wird getroffen, indem beobachtet und festgestellt wird, wie oft Tabellen modifiziert werden und inwieweit sich die Tabellenstatistiken geändert haben. Der Algorithmus für die automatische Statistikerfassung lernt mit der Zeit, wie schnell sich die Statistiken für die einzelnen Tabellen ändern und terminiert intern eine entsprechende Ausführung des Dienstprogramms RUNSTATS.

Die normalen Datenbankverwaltungsaktivitäten, zum Beispiel wenn ein Benutzer die Dienstprogramme RUNSTATS oder REORG ausführt oder eine Tabelle ändert oder löscht, werden von einer Aktivierung dieser Funktion nicht berührt.

Wenn Sie sich nicht sicher sind, wie häufig Statistiken für die Tabellen in Ihrer Datenbank zu erfassen sind, können Sie die Funktion zur automatischen Statistikerfassung in Ihren Gesamtplan zur Datenbankverwaltung aufnehmen.

Die Funktion zur automatischen Statistikerfassung kann mit Hilfe der Konfigurationsparameter **auto_runstats**, **auto_tbl_maint** und **auto_maintdatabase** aktiviert bzw. inaktiviert werden.

In einer Umgebung mit mehreren Datenbankpartitionen erfolgt die Festlegung, eine automatische Statistikerfassung auszuführen, und die Einleitung der automatischen Statistikerfassung in der Katalogtabellenpartition. Der Konfigurationsparameter **auto_runstats** muss nur in der Katalogtabellenpartition aktiviert werden. Die tatsächliche Statistikerfassung wird durch das Dienstprogramm RUNSTATS wie folgt ausführt:

1. Wenn die Katalogtabellenpartition Tabellendaten enthält, werden Statistiken in der Katalogtabellenpartition erfasst. Das Dienstprogramm RUNSTATS erfasst immer Statistiken in der Partition, in der es eingeleitet wird, sofern diese Partition Tabellendaten enthält.
2. Ansonsten erfolgt die Erfassung von Statistiken in der ersten Partition der Tabellenpartitionsliste.

Tabellen, die für die automatische Statistikerfassung in Betracht kommen, können über den Assistenten zur automatischen Verwaltung in der Steuerzentrale oder der Diagnosezentrale konfiguriert werden.

Automatische Statistikprofilerstellung:

Fehlende oder veraltete Statistiken können dazu führen, dass das Optimierungsprogramm einen langsameren Abfrageplan auswählt. Es ist jedoch wichtig zu beachten, dass nicht alle Statistiken für eine bestimmte Auslastung eine wesentliche Rolle spielen. Zum Beispiel haben Statistiken für Spalten, die in keinem Abfragevergleichselement auftreten, mit hoher Wahrscheinlichkeit keinen Einfluss. Einige Statistiken über mehrere Spalten (Spaltengruppenstatistiken) werden benötigt, um Korrelationen zwischen diesen Spalten zu unterstützen.

Die automatische Statistikprofilerstellung analysiert das Verhalten des Optimierungsprogramms, indem sie nur Spalten betrachtet, die in vorherigen Abfragen verwendet wurden, und Spalten oder Spaltenkombinationen aufzeichnet,

bei denen Schätzfehler aufgetreten sind. Zur Erkennung von Fehlern und zur Empfehlung bzw. Änderung eines statistischen Profils untersucht der Statistikprofilgenerator die Informationen, die beim Kompilieren der Abfrage gesammelt werden, und die Informationen, die bei der Ausführung der Abfrage aufgezeichnet werden. Dieses Verfahren arbeitet rückwirkend, da die Aktion ausgeführt wird, nachdem die Abfrage geprüft und schließlich ein Plan ausgewählt und ausgeführt wurde.

Die automatische Statistikprofilerstellung gibt Empfehlungen, wie Statistiken mit Hilfe des Dienstprogramms RUNSTATS zu erfassen sind, indem sie veraltete, fehlende und falsch spezifizierte Statistikdaten erkennt und Statistikprofile auf der Basis des Abfragefeedbacks generiert.

Wenn dies Ihren Anforderungen entgegenkommt, können Sie die Funktion zur automatischen Statistikprofilerstellung in Ihren Gesamtplan zur Datenbankverwaltung aufnehmen.

Die automatische Statistikprofilerstellung interagiert mit der automatischen Statistikerfassung und gibt Empfehlungen, wann Statistiken erfasst werden sollten.

Die Funktion zur automatischen Statistikprofilerstellung kann mit Hilfe der Konfigurationsparameter **auto_stats_prof**, **auto_tbl_maint** und **auto_maintdatabase** aktiviert bzw. inaktiviert werden. Wenn der Datenbankkonfigurationsparameter **auto_prof_upd** ebenfalls aktiviert wird, werden die generierten Statistikprofile zur Aktualisierung der RUNSTATS-Benutzerprofile verwendet. Die automatische Statistikprofilerstellung steht für Umgebungen mit mehreren Datenbankpartitionen oder für Systeme mit aktivierter partitionsübergreifender Parallelität (Parallelität in symmetrischen Mehrprozessorsystemen (SMP-Systemen)) nicht zur Verfügung.

Automatische Reorganisation:

Nach zahlreichen Änderungen an Tabellendaten befinden sich logisch sequenzielle Daten vielleicht auf nicht sequenziellen physischen Seiten, so dass der Datenbankmanager für den Datenzugriff zusätzliche Leseoperationen ausführen muss.

Die vom Dienstprogramm RUNSTATS gesammelten Statistikinformationen zeigen unter anderem die Datenverteilung innerhalb einer Tabelle. Insbesondere kann durch eine Analyse dieser Statistiken ermittelt werden, wann eine Reorganisation und welche Art der Reorganisation erforderlich ist. Die automatische Reorganisation bestimmt die Erforderlichkeit einer Reorganisation für Tabellen mit Hilfe der REORGCHK-Formeln. Sie werten in regelmäßigen Abständen Tabellen aus, deren Statistiken aktualisiert wurden, um zu prüfen, ob eine Reorganisation erforderlich ist. Wenn dies der Fall ist, terminiert sie intern eine klassische Reorganisation für die Tabelle. Dies setzt voraus, dass Ihre Anwendungen ohne Schreibzugriff auf die Tabellen arbeiten, die reorganisiert werden.

Die Funktion zur automatischen Reorganisation kann mit Hilfe der Datenbankkonfigurationsparameter **auto_reorg**, **auto_tbl_maint** und **auto_maint** aktiviert bzw. inaktiviert werden.

In einer Umgebung mit mehreren Datenbankpartitionen erfolgt die Festlegung, eine automatische Reorganisation auszuführen, und die Einleitung der automatischen Reorganisation in der Katalogtabellenpartition. Die Datenbankkonfigurationsparameter müssen lediglich in der Katalogtabellenpartition aktiviert werden. Die Reorganisation wird in allen Partitionen ausgeführt, in denen sich die Zieltabellen befinden.

Wenn Sie sich nicht sicher sind, wann und wie Ihre Tabellen und Indizes zu reorganisieren sind, können Sie die automatische Reorganisation in Ihren Gesamtplan zur Datenbankverwaltung aufnehmen.

Tabellen, die für die automatische Reorganisation in Betracht kommen, können über den Assistenten zur automatischen Verwaltung in der Steuerzentrale oder der Diagnosezentrale konfiguriert werden.

Verwaltungsfenster zur Automatisierung:

Die oben beschriebenen automatischen Verwaltungsfunktionen beanspruchen Ressourcen auf Ihrem System und können bei der Ausführung die Leistung Ihrer Datenbank beeinträchtigen. Die automatische Reorganisation und die Offlinedatenbanksicherung schränken während der Ausführung zudem den Zugriff auf die Tabellen und die Datenbank ein. Daher ist es erforderlich, geeignete Zeiträume vorzusehen, in denen diese Verwaltungsaktivitäten intern zur Ausführung durch DB2 UDB terminiert werden können. Diese können als Zeiträume für Offline- oder Onlineverwaltung im Assistenten zur automatischen Verwaltung über die Steuerzentrale oder die Diagnosezentrale angegeben werden.

Offlinedatenbanksicherungen und Tabellen- und Indexreorganisationen werden im Zeitraum für die Offlineverwaltung ausgeführt. Diese Funktionen werden bis zum Ende ausgeführt, selbst wenn sie dabei den angegebenen Zeitraum überschreiten. Der Mechanismus zur internen Terminierung lernt mit der Zeit und schätzt die Zeiten für die Jobausführung ab. Wenn der Offlinezeitraum für eine bestimmte Datenbanksicherung oder eine Reorganisationsaktivität zu klein ist, startet der Scheduler den Job beim nächsten Mal nicht wieder und überlässt es dem Diagnosemonitor, eine Benachrichtigung über eine erforderliche Erweiterung des Zeitraums für die Offlineverwaltung auszugeben.

Die automatische Statistikerfassung und Statistikprofilerstellung sowie Onlinedatenbanksicherungen werden im Zeitraum für die Onlineverwaltung ausgeführt. Zur Minimierung der Auswirkungen auf das System werden sie durch einen adaptiven Drosselmechanismus für Dienstprogramme gedrosselt. Der interne Terminierungsmechanismus verwendet den Onlineverwaltungszeitraum zum Starten der Onlinejobs. Diese Funktionen werden bis zum Ende ausgeführt, selbst wenn sie dabei den angegebenen Zeitraum überschreiten.

Speicher:

Die automatischen Funktionen zur Statistikerfassung und Reorganisation speichern Arbeitsdaten in Tabellen Ihrer Datenbank. Diese Tabellen werden im Tabellenbereich SYSTOOLSPACE erstellt. Der Tabellenbereich SYSTOOLSPACE wird automatisch mit Standardoptionen erstellt, wenn die Datenbank aktiviert wird. Der Speicherbedarf für diese Tabellen ist proportional zur Anzahl der Tabellen in der Datenbank und sollte mit ca. 1 KB pro Tabelle veranschlagt werden. Wenn dies für Ihre Datenbank eine beträchtliche Größe ist, kann es sinnvoll sein, den Tabellenbereich selbst zu löschen und erneut zu erstellen, um eine angemessene Speicherkapazität zuzuordnen. Die Tabellen der automatischen Verwaltung und des Diagnosemonitors in diesem Tabellenbereich werden automatisch erneut erstellt. Alle erfassten älteren Daten in diesen Tabellen gehen verloren, wenn der Tabellenbereich gelöscht wird.

Überwachung und Benachrichtigung:

Der Diagnosemonitor stellt Funktionalität zur Überwachung und Benachrichtigung für die automatischen Funktionen der Datenbanksicherung, Statistikerfassung und Reorganisation zur Verfügung.

Zugehörige Konzepte:

- „Katalogstatistiken" in *Systemverwaltung: Optimierung*
- „Tabellenreorganisation" in *Systemverwaltung: Optimierung*
- „Tabellen- und Indexverwaltung für Standardtabellen" in *Systemverwaltung: Optimierung*
- „Entwickeln einer Sicherungs- und Wiederherstellungsstrategie" auf Seite 17
- „Tabellen- und Indexverwaltung für MDC-Tabellen" in *Systemverwaltung: Optimierung*
- „Health monitor" in *System Monitor Guide and Reference*
- „Automatische Statistikerfassung" in *Systemverwaltung: Optimierung*

Zugehörige Referenzen:

- „autonomic_switches - Schalter für automatische Verwaltung (Konfigurationsparameter)" in *Systemverwaltung: Optimierung*

Übersicht über die HADR-Funktion (High Availability Disaster Recovery)

Die HADR-Funktion (High Availability Disaster Recovery) von DB2® Universal Database (DB2 UDB) ist eine Einrichtung zur Datenbankreplikation, die eine Lösung zur Realisierung hoher Verfügbarkeit sowohl für Teilausfälle als auch für Gesamtausfälle von Standorten zur Verfügung stellt. HADR schützt gegen Datenverlust durch eine Replikation von Datenänderungen aus einer Quellendatenbank, die als *Primärdatenbank* bezeichnet wird, in eine Zieldatenbank, die als *Bereitschaftsdatenbank* bezeichnet wird.

Ein Teilausfall eines Standorts kann durch einen Fehler in der Hardware, im Netzwerk oder in der Software (DB2 UDB bzw. Betriebssystem) verursacht werden. Ohne HADR muss der Server des Datenbankmanagementsystems (DBMS) bzw. die Maschine, auf der sich die Datenbank befindet, erneut gebootet oder gestartet werden. Dieser Prozess kann einige Minuten in Anspruch nehmen. Mit HADR kann die Bereitschaftsdatenbank die Funktion der Primärdatenbank innerhalb weniger Sekunden übernehmen.

Ein Komplettausfall eines Standorts kann auftreten, wenn eine Katastrophe, zum Beispiel ein Brand, den gesamten Standort zerstört. Da HADR mit TCP/IP zur Kommunikation zwischen der Primär- und der Bereitschaftsdatenbank arbeitet, können die beiden Datenbanken an verschiedenen Standorten stationiert werden. Zum Beispiel könnte sich Ihre Primärdatenbank am Hauptsitz in einer Stadt befinden, während sich die Bereitschaftsdatenbank in einer Vertriebsniederlassung in einer anderen Stadt befindet. Wenn ein Katastrophenfall am Hauptstandort eintritt, bleibt die Datenverfügbarkeit dadurch erhalten, das die ferne Bereitschaftsdatenbank die Funktion als Primärdatenbank mit vollständiger DB2 UDB-Funktionalität übernimmt.

Nachdem die Übernahmeoperation erfolgt ist, können Sie die ursprüngliche Primärdatenbank wieder betriebsbereit machen und in ihren Status als Primärdatenbank zurückversetzen. Dies wird als Zurückübertragung (*failback*) bezeichnet.

Mit HADR können Sie die gewünschte Stufe des Schutzes gegen potenziellen Datenverlust wählen, indem Sie einen von drei Synchronisationsmodi auswählen: Synchron (SYNC), Fast synchron (NEARSYNC) und Asynchron (ASYNC). Diese Modi geben an, wie Datenänderungen zwischen den beiden Systemen weitergegeben werden. Der ausgewählte Synchronisationsmodus legt fest, wie nahe die Bereitschaftsdatenbank an ein exaktes Replikat der Primärdatenbank heranreicht. Im synchronen Modus kann HADR zum Beispiel sicherstellen, dass jede Transaktion, die in der Primärdatenbank festgeschrieben wird, auch in der Bereitschaftsdatenbank festgeschrieben wird.

Die Synchronisation ermöglicht eine Funktionsübernahme und Zurückübertragung der Funktion zwischen den beiden Systemen.

Datenänderungen werden in Protokollsätzen der Datenbank aufgezeichnet, die anschließend vom Primärsystem an das Bereitschaftssystem gesendet werden. HADR ist eng mit der Protokollierung und Wiederherstellung von DB2 UDB gekoppelt.

HADR setzt jedoch voraus, dass beide System mit der gleichen Hardware, dem gleichen Betriebssystem und der gleichen DB2 UDB-Software ausgestattet sind. (Es kann kleinere Unterschiede während der Zeiträume geben, in denen die Systeme durch Upgrades aufgerüstet werden.)

Die HADR-Bereitschaftsdatenbank wird eingerichtet, indem sie von einer Sicherungskopie der Primärdatenbank wiederhergestellt oder durch eine geteilte Spiegelkopie der Primärdatenbank initialisiert wird. Wenn HADR einmal gestartet ist, ruft die Bereitschaftsdatenbank Protokollsätze aus der Primärdatenbank ab und vollzieht die protokollierten Änderungen an der eigenen Kopie der Datenbank nach. Die Protokollsätze werden auf die Bereitschaftsdatenbank angewendet, bis die Bereitschaftsdatenbank die Primärdatenbank bis zur momentan im Speicher befindlichen Protokollgruppe „eingeholt" hat. An diesem Punkt geht das HADR-Paar in den PEER-Status über, in dem die Primärdatenbank neue Protokollseiten an die Bereitschaftsdatenbank sendet und die Seiten gleichzeitig auf ihre lokale Platte schreibt. Die Protokollseiten werden in der Bereitschaftsdatenbank sofort nach ihrem Eintreffen nachvollzogen. Durch das kontinuierliche Nachvollziehen der Protokolle wird die Bereitschaftsdatenbank als zeitverzögertes Replikat der Primärdatenbank gepflegt.

Wenn es in der Primärdatenbank zu einem Ausfall kommt, kann die Bereitschaftsdatenbank problemlos die Funktion übernehmen (Failover). Wenn die Bereitschaftsdatenbank die Funktion übernommen hat, wird sie zur neuen Primärdatenbank. Da der Bereitschaftsdatenbankserver bereits online ist, kann die Funktionsübernahme sehr rasch ausgeführt werden. Dadurch wird die Ausfallzeit der Datenbankaktivität auf ein Minimum reduziert.

HADR kann außerdem genutzt werden, um die Datenbankverfügbarkeit über einen Zeitraum bestimmter Release-Upgrades für Hard- und Software sicherzustellen. Sie können Upgrades für die Hardware-, Betriebssystem- oder DB2 UDB Fix-Pak-Stufe des Bereitschaftssystems durchführen, während die Primärdatenbank für Anwendungen verfügbar ist. Anschließend können Sie die Anwendungen auf das aufgerüstete System übertragen, während die Upgrades auf dem Primärsystem durchgeführt werden.

Die Leistung der neuen Primärdatenbank unmittelbar nach der Funktionsübernahme ist möglicherweise nicht mit der der Primärdatenbank vor dem Ausfall identisch. Die neue Primärdatenbank benötigt etwas Zeit, um den Anweisungscache, den Pufferpool und andere Speicherpositionen zu füllen, die durch den Datenbankmanager verwendet werden. Obwohl das Nachvollziehen der Protokolldaten aus der alten Primärdatenbank Daten zum Teil in den Pufferpool und die Systemkatalogcaches versetzt, geschieht dies nicht vollständig, weil es nur auf Schreibaktivitäten zurückzuführen ist. Häufig genutzte Indexseiten, Kataloginformationen für Tabellen, die abgefragt, jedoch nicht aktualisiert werden, Anweisungscaches und Zugriffspläne sind sämtlich noch nicht in den Cachespeichern vorhanden. Der gesamte Prozess läuft indes schneller ab, als wenn Sie eine neue DB2 UDB-Datenbank starten würden.

Wenn der frühere Primärserver repariert ist, kann er als Bereitschaftsdatenbank reintegriert werden, wenn die beiden Kopien der Datenbank wieder konsistent gemacht werden können. Nach der Reintegration kann eine Zurückübertragung der Funktion durchgeführt werden, so dass die ursprüngliche Primärdatenbank wieder zur Primärdatenbank wird.

Die HADR-Funktion steht nur unter DB2 UDB Enterprise Server Edition (ESE) zur Verfügung. In anderen Editionen wie Personal Edition und ESE mit Datenbankpartitionierungsfunktion (DPF) ist sie inaktiviert.

HADR wird auf der Datenbankebene, nicht der Exemplarebene ausgeführt. Dies bedeutet, dass ein einziges Exemplar eine Primärdatenbank (A), eine Bereitschaftsdatenbank (B) und eine Standarddatenbank (C) ohne HADR enthalten könnte. Allerdings kann ein Exemplar nicht sowohl die Primärkopie als auch die Bereitschaftskopie derselben Datenbank enthalten, da HADR voraussetzt, dass beide Kopien der Datenbank denselben Datenbanknamen besitzen.

Zugehörige Konzepte:

- „Hohe Verfügbarkeit" in *Datenwiederherstellung und hohe Verfügbarkeit Handbuch und Referenz*

Sicherheit

Zum Schutz von Daten und Ressourcen auf einem Datenbankserver verwendet DB2® Universal Database eine Kombination aus externen Sicherheitsservices und internen Zugriffssteuerinformationen. Wenn Sie auf den Datenbankserver zugreifen wollen, müssen Sie einige Sicherheitsüberprüfungen durchlaufen, bevor Ihnen Zugriff auf Datenbankdaten oder Ressourcen gewährt wird. Der erste Schritt in der Datenbanksicherheit wird als *Authentifizierung* (Identitätsüberprüfung) bezeichnet, bei der ein Benutzer nachweisen muss, dass er der ist, der er behauptet zu sein. Der zweite Schritt wird als *Berechtigung* (Authorization) bezeichnet, bei der der Datenbankmanager entscheidet, ob ein überprüfter Benutzer die angeforderte Aktion ausführen oder auf die angeforderten Daten zugreifen darf.

Zugehörige Konzepte:

- „Authentifizierung" auf Seite 29
- „Berechtigung" auf Seite 30

Authentifizierung

Die Authentifizierung (Identifikationsüberprüfung) wird mit Hilfe einer Sicherheitseinrichtung außerhalb von DB2® Universal Database (DB2 UDB) vollzogen. Die Sicherheitseinrichtung kann Teil des Betriebssystems, ein separates Produkt oder, in bestimmten Fällen, auch nicht vorhanden sein. Auf UNIX®-basierten Systemen ist die Sicherheitseinrichtung in das Betriebssystem selbst integriert. Die Sicherheitseinrichtung benötigt zwei Elemente zur Authentifizierung eines Benutzers: eine Benutzer-ID und ein Kennwort. Die Benutzer-ID identifiziert den Benutzer der Sicherheitseinrichtung gegenüber. Durch die Eingabe des richtigen Kennworts (eine Information, die nur dem Benutzer und der Sicherheitseinrichtung bekannt ist) wird die Identität des Benutzers (entsprechend der Benutzer-ID) bestätigt.

Nach erfolgreicher Authentifizierung:

- Der Benutzer muss für DB2 UDB mit Hilfe eines SQL-Berechtigungsnamens oder einer Berechtigungs-ID (*authid*) identifiziert werden. Der Name kann mit der Benutzer-ID übereinstimmen oder ein zugeordneter Wert sein. Beispielsweise wird auf UNIX-basierten Systemen eine DB2 UDB-Berechtigungs-ID (*authid*) abgeleitet, indem eine UNIX-Benutzer-ID, die den DB2 UDB-Namenskonventionen entspricht, in Großbuchstaben umgesetzt wird.
- Eine Liste von Gruppen, zu denen der Benutzer gehört, wird abgerufen. Die Gruppenzugehörigkeit kann bei der Berechtigung des Benutzers verwendet werden. Gruppen sind Definitionseinheiten von Sicherheitseinrichtungen, denen auch DB2 UDB-Berechtigungsnamen zugeordnet sein müssen. Diese Zuordnung erfolgt mit einer ähnlichen Methode wie für Benutzer-IDs.

DB2 UDB verwendet die Sicherheitseinrichtung zur Authentifizierung von Benutzern auf eine von zwei Arten:

- DB2 UDB verwendet eine erfolgreiche Anmeldung durch das Sicherheitssystem als Nachweis der Identität und lässt Folgendes zu:
  - Verwendung lokaler Befehle zum Zugriff auf lokale Daten
  - Verwendung von Fernverbindungen, bei denen der Server die Authentifizierung auf dem Client akzeptiert
- DB2 UDB akzeptiert eine Kombination aus Benutzer-ID und Kennwort. DB2 verwendet eine erfolgreiche Gültigkeitsprüfung dieses Paares durch die Sicherheitseinrichtung als Identitätsnachweis und erlaubt Folgendes:
  - Verwendung von Fernverbindungen, bei denen der Server einen Beweis der Authentifizierung verlangt
  - Verwendung von Operationen, bei denen der Benutzer einen Befehl unter einer anderen als der zur Anmeldung verwendeten Identität ausführen will

Mit DB2 UDB unter AIX® können fehlgeschlagene Kennworteingaben beim Betriebssystem protokolliert werden. Dadurch lässt sich ermitteln, wann ein Client die Anzahl zulässiger Anmeldeversuche überschritten hat, die mit dem Parameter LOGINRETRIES festgelegt wird.

Zugehörige Konzepte:

- „Authentifizierungsmethoden für den Server" in *Systemverwaltung: Implementierung*
- „Zugriffsrechte, Berechtigungsstufen und Datenbankberechtigungen" in *Systemverwaltung: Implementierung*
- „Sicherheit" auf Seite 28
- „Berechtigung" auf Seite 30

Berechtigung

Die Berechtigung (Authorization) ist der Prozess, durch den DB2® Informationen über einen authentifizierten DB2-Benutzer abruft, die angeben, welche Datenbankoperationen der Benutzer durchführen und auf welche Datenobjekte er zugreifen darf. Bei jeder Benutzeranforderung kann es mehr als eine Berechtigungsprüfung geben, je nachdem, welche Objekte und Operationen beteiligt sind.

Die Berechtigung wird mit Hilfe von DB2-Einrichtungen durchgeführt. DB2-Tabellen und DB2-Konfigurationsdateien werden zur Speicherung der Berechtigungen verwendet, die jedem Berechtigungsnamen zugeordnet sind. Der Berechtigungsname eines authentifizierten Benutzers und die Namen der Gruppen, zu denen der Benutzer gehört, werden mit den gespeicherten Berechtigungen verglichen. Anhand dieses Vergleichs entscheidet DB2, ob der angeforderte Zugriff gewährt wird.

Es gibt zwei Arten der Berechtigungen, die von DB2 Universal Database™ (DB2 UDB) gespeichert werden: Zugriffsrechte und Berechtigungsstufen. Ein *Zugriffsrecht* definiert eine einzelne Berechtigung für einen Berechtigungsnamen, Datenbankressourcen zu erstellen oder auf sie zuzugreifen. Zugriffsrechte werden in den Datenbankkatalogen gespeichert. *Berechtigungsstufen* bilden eine Methode, Zugriffsrechte und Kontrolle über Operationen zur Wartung des Datenbankmanagers und über Dienstprogrammoperationen auf einer höheren Stufe in Gruppen zusammenzufassen. Datenbankspezifische Berechtigungen werden in den Datenbankkatalogen gespeichert, während Systemberechtigungen durch Gruppenzugehörigkeit zugeordnet und die den Berechtigungsstufen zugeordneten Gruppennamen in der Konfigurationsdatei des Datenbankmanagers für das jeweilige Exemplar gespeichert werden.

Gruppen stellen eine zweckmäßige Methode dar, für einen Benutzerverbund Berechtigungen durchzuführen, ohne jedem Benutzer einzeln Zugriffsrechte erteilen bzw. entziehen zu müssen. Sofern nicht anders angegeben, können Gruppenberechtigungsnamen überall dort verwendet werden, wo Berechtigungsnamen zu Berechtigungszwecken verwendet werden. Im Allgemeinen wird die Gruppenzugehörigkeit für dynamisches SQL und Berechtigungen für Nichtdatenbankobjekte (z. B. Befehle auf Exemplarebene und Dienstprogramme) und nicht für statisches SQL berücksichtigt. Ein Ausnahmefall zu diesem allgemeinen Fall ist das Erteilen von Zugriffsrechten für PUBLIC, die bei der Verarbeitung von statischem SQL berücksichtigt werden. Spezielle Fälle, in denen Gruppenzugehörigkeit nicht gültig ist, werden in der DB2 UDB-Dokumentation entsprechend vermerkt.

Zugehörige Konzepte:

- „Authorization and privileges" in *SQL Reference, Volume 1*
- „Zugriffsrechte, Berechtigungsstufen und Datenbankberechtigungen" in *Systemverwaltung: Implementierung*
- „Sicherheit" auf Seite 28

Arbeitseinheiten

Eine Transaktion wird in DB2® Universal Database (DB2 UDB) allgemein als *Arbeitseinheit* bezeichnet. Eine Arbeitseinheit ist eine wiederherstellbare Folge von Operationen innerhalb eines Anwendungsprozesses. Der Datenbankmanager kann mit ihrer Hilfe sicherstellen, dass sich eine Datenbank in einem konsistenten Status befindet. Jeder Lese- oder Schreibvorgang in der Datenbank erfolgt innerhalb einer Arbeitseinheit.

Eine Banktransaktion könnte beispielsweise darin bestehen, dass eine Geldsumme von einem Sparkonto auf ein Girokonto überwiesen wird. Nachdem die Anwendung einen Betrag vom Sparkonto abgezogen hat, sind die beiden Konten inkonsistent und bleiben es, bis der Betrag auf dem Girokonto gutgeschrieben ist. Wenn *beide* Schritte zu Ende geführt sind, ist ein Konsistenzzustand erreicht. Die Änderungen können festgeschrieben und anderen Anwendungen zur Verfügung gestellt werden.

Eine Arbeitseinheit beginnt, wenn die erste SQL-Anweisung für die Datenbank ausgeführt wird. Die Anwendung muss die Arbeitseinheit beenden, indem sie die Anweisung COMMIT oder ROLLBACK absetzt. Die Anweisung COMMIT schreibt alle Änderungen, die innerhalb der Arbeitseinheit vorgenommen wurden, permanent fest. Die Anweisung ROLLBACK löscht diese Änderungen aus der Datenbank. Wenn bei einer normalen Beendigung der Anwendung keine dieser Anweisungen abgesetzt wird, wird die Arbeitseinheit automatisch festgeschrieben. Wenn die Anwendung inmitten einer Arbeitseinheit abnormal beendet wird, wird die Arbeitseinheit automatisch rückgängig gemacht. Sobald eine Anweisung COMMIT oder ROLLBACK abgesetzt wurde, kann sie nicht mehr gestoppt werden. Bei einigen Multithread-Anwendungen oder Betriebssystemen (z. B. Windows®) wird die Arbeitseinheit, wenn eine Anwendung normal, jedoch ohne explizite Ausführung einer dieser Anweisungen beendet wird, automatisch rückgängig (ROLLBACK) gemacht. Es wird daher empfohlen, dafür zu sorgen, dass Anwendungen vollständige Arbeitseinheiten immer explizit mit einer Anweisung COMMIT festschreiben oder mit einer Anweisung ROLLBACK rückgängig machen. Wenn ein Teil einer Arbeitseinheit nicht erfolgreich beendet wird, werden die Aktualisierungen rückgängig gemacht, so dass die beteiligten Tabellen in dem Zustand verbleiben, in dem sie vor Beginn der Transaktion waren. Auf diese Weise wird sichergestellt, dass Anforderungen weder verloren gehen noch mehrfach ausgeführt werden.

Zugehörige Referenzen:

- „COMMIT statement" in *SQL Reference, Volume 2*
- „ROLLBACK statement" in *SQL Reference, Volume 2*

Kapitel 2. Parallele Datenbanksysteme

Datenpartitionierung

DB2® Universal Database (DB2 UDB) erweitert den Datenbankmanager auf die parallele Mehrknotenumgebung. Eine *Datenbankpartition* ist ein Teil einer Datenbank, der aus seinen eigenen Daten, Indizes, Konfigurationsdateien und Transaktionsprotokollen besteht. Eine Datenbankpartition wird manchmal auch als Knoten oder Datenbankknoten bezeichnet.

Eine *Einzelpartitionsdatenbank* ist eine Datenbank, die nur über eine Datenbankpartition verfügt. Sämtliche Daten in der Datenbank werden in dieser Partition gespeichert. In diesem Fall bieten Datenbankpartitionsgruppen, wenn sie vorhanden sind, keine zusätzlichen Leistungsvorteile.

Eine *partitionierte Datenbank* ist eine Datenbank mit zwei oder mehr Datenbankpartitionen. Die Tabellen können in einer oder mehreren Datenbankpartitionen gespeichert werden. Wenn eine Tabelle in einer Datenbankpartitionsgruppe mit mehreren Partitionen gespeichert ist, heißt dies, dass einige ihrer Zeilen in einer Partition gespeichert werden, während sich andere Zeilen in anderen Partitionen befinden.

In der Regel existiert eine einzelne Datenbankpartition auf jedem physischen Knoten, und die Prozessoren auf jedem System werden vom Datenbankmanager in jeder Datenbankpartition zur Verwaltung des jeweiligen Teils der Gesamtdaten in der Datenbank verwendet.

Da die Daten auf Partitionen aufgeteilt sind, können Sie das Potenzial mehrerer Prozessoren auf mehreren physischen Knoten zur Erfüllung von Informationsanforderungen nutzen. Anforderungen zur Abfrage und Aktualisierung von Daten werden automatisch in Unteranforderungen zerlegt und in den betroffenen Datenbankpartitionen parallel ausgeführt. Die Tatsache, dass Datenbanken auf Datenbankpartitionen verteilt sind, ist für Benutzer, die SQL-Anweisungen ausführen, transparent.

Die Benutzerinteraktion erfolgt über nur eine Datenbankpartition, die als *Koordinatorknoten* für den jeweiligen Benutzer bezeichnet wird. Der Koordinatorknoten ist in derselben Datenbankpartition aktiv wie die Anwendung bzw., im Fall einer fernen Anwendung, in der Datenbankpartition, mit der die Anwendung verbunden ist. Eine beliebige Datenbankpartition kann als Koordinatorknoten verwendet werden.

DB2 UDB unterstützt ein partitioniertes Speichermodell, das Ihnen ermöglicht, Daten über verschiedene Datenbankpartitionen in der Datenbank verteilt zu speichern. Dies bedeutet, dass die Daten zwar physisch über mehr als eine Datenbankpartition verteilt werden, jedoch so auf sie zugegriffen werden kann, als ob sie sich an derselben Speicherposition befänden. Anwendungen und Benutzer, die auf Daten in einer partitionierten Datenbank zugreifen, brauchen die physische Speicherposition der Daten nicht zu kennen.

Die Daten werden trotz der physischen Trennung als logische Einheit verwendet und verwaltet. Benutzer können wählen, wie ihre Daten partitioniert werden,

indem Sie Partitionierungsschlüssel deklarieren. Benutzer können darüber hinaus bestimmen, auf welche und auf wie viele Datenbankpartitionen ihre Tabellendaten verteilt werden können, indem sie den Tabellenbereich und die zugeordnete Datenbankpartitionsgruppe auswählen, in der die Daten gespeichert werden sollen. Empfehlungen zur Partitionierung und Replikation können mit Hilfe von DB2 Designadvisor ermittelt werden. Weiterhin wird eine aktualisierbare Partitionierungszuordnung zusammen mit einem Hashalgorithmus verwendet, um die Zuordnung von Werten der Partitionierungsschlüssel zu Datenbankpartitionen anzugeben, wodurch das Platzieren und Abrufen der einzelnen Datenzeilen festgelegt wird. Dadurch können Sie die Auslastung für große Tabellen über eine partitionierte Datenbank hinweg verteilen, während kleinere Tabellen in einer oder mehreren Datenbankpartitionen gespeichert werden können. Jede Datenbankpartition verfügt über lokale Indizes für die in ihr gespeicherten Daten, so dass für den lokalen Datenzugriff eine erhöhte Leistung erzielt wird.

Sie sind nicht darauf beschränkt, alle Tabellen über alle Datenbankpartitionen in der Datenbank verteilt zu speichern. DB2 UDB unterstützt eine *Teilentclusterung*, das heißt, Sie können die Tabellen und die zugehörigen Tabellenbereiche auf eine Untergruppe der Datenbankpartitionen im System verteilen.

Eine alternative Methode, die Sie in Erwägung ziehen können, wenn Sie Tabellen in den einzelnen Datenbankpartitionen angelegt haben wollen, ist die Verwendung und Replikation gespeicherter Abfragetabellen. Sie können eine gespeicherte Abfragetabelle mit den von Ihnen benötigten Informationen erstellen und diese dann auf jeden Knoten replizieren.

Parallelität

Komponenten einer Task, wie zum Beispiel einer Datenbankabfrage, können parallel ausgeführt werden, um die Leistung erheblich zu erhöhen. Die Art der Task, die Datenbankkonfiguration und die Hardwareumgebung bestimmen, wie DB2® Universal Database (DB2 UDB) eine Task parallel ausführt. Diese Aspekte stehen in Wechselbeziehung zueinander und sollten bei der Arbeit am physischen und logischen Entwurf einer Datenbank im Zusammenhang betrachtet werden. DB2 UDB unterstützt die folgenden Arten von Parallelität:

- Ein-/Ausgabeparallelität
- Abfrageparallelität
- Dienstprogrammparallelität

Ein-/Ausgabeparallelität

Wenn mehrere Behälter für einen Tabellenbereich vorhanden sind, kann der Datenbankmanager die *parallele Ein-/Ausgabe* nutzen. Parallele E/A bezeichnet den Vorgang, bei dem Lese- bzw. Schreiboperationen mit zwei oder mehr Ein-/Ausgabeeinheiten gleichzeitig durchgeführt werden. Auf diese Weise können Verbesserungen des Durchsatzes erzielt werden.

Abfrageparallelität

Es gibt zwei Arten der Abfrageparallelität: abfrageübergreifende und abfrageinterne Parallelität.

Abfrageübergreifende Parallelität bezeichnet die Fähigkeit einer Datenbank, Abfragen von mehreren Anwendungen gleichzeitig zu akzeptieren. Jede Abfrage wird unabhängig von den anderen ausgeführt, DB2 UDB führt jedoch alle Abfragen gleichzeitig aus. DB2 UDB unterstützt diese Art der Parallelität seit jeher.

Abfrageinterne Parallelität bezeichnet die gleichzeitige Verarbeitung von Teilen einer einzelnen Abfrage mit Hilfe der *partitionsinternen Parallelität* bzw. der *partitionsübergreifenden Parallelität* (oder beiden).

Partitionsinterne Parallelität

Der Begriff *partitionsinterne Parallelität* bezeichnet die Fähigkeit, eine Abfrage in mehrere Teile zu untergliedern. Einige DB2 UDB-Dienstprogramme arbeiten ebenfalls mit dieser Art der Parallelität.

Bei der partitionsinternen Parallelität wird das, was im Allgemeinen als eine einzige Datenbankoperation betrachtet wird (z. B. eine Indexerstellung, das Laden von Daten, SQL-Abfragen) in mehrere Teile unterteilt, von denen viele oder alle parallel *innerhalb einer einzigen Datenbankpartition* ausgeführt werden können.

Abb. 13 zeigt eine Abfrage, die in vier Teile aufgeteilt ist, die parallel ausgeführt werden können, wobei die Ergebnisse schneller geliefert werden als bei serieller Ausführung der Abfrage. Die Teile sind jeweils Kopien voneinander. Zur Nutzung der partitionsinternen Parallelität muss die Datenbank entsprechend konfiguriert werden. Sie können den Grad der Parallelität selbst auswählen oder ihn vom System festlegen lassen. Der Grad der Parallelität stellt die Anzahl der Teile einer Abfrage dar, die parallel ausgeführt werden.

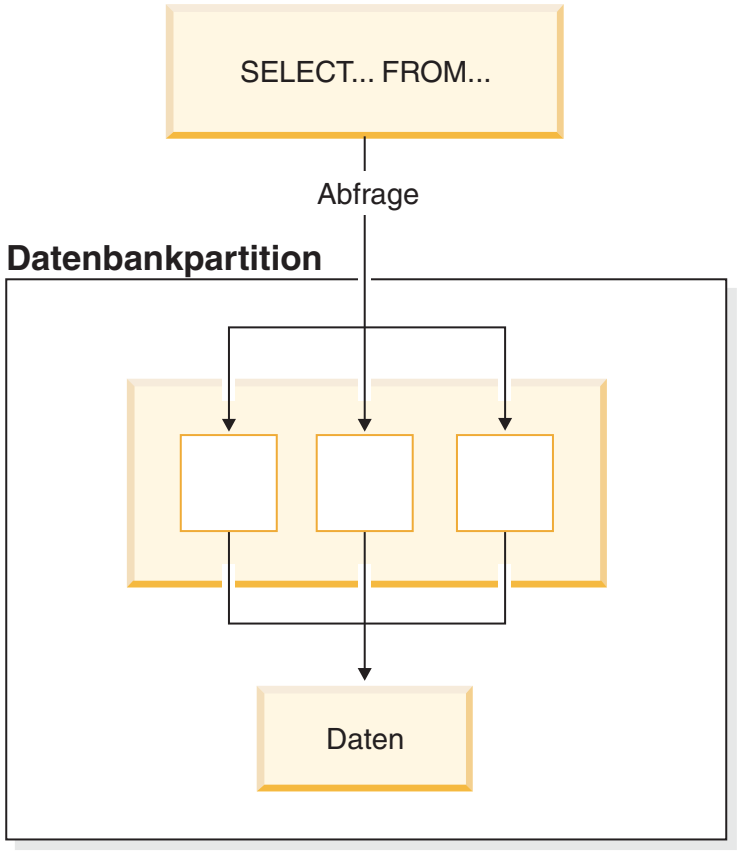


Abbildung 13. Partitionsinterne Parallelität

Partitionsübergreifende Parallelität

Der Begriff *partitionsübergreifende Parallelität* bezeichnet die Fähigkeit, eine Abfrage in mehreren Teilen über mehrere Partitionen einer partitionierten Datenbank auf einer oder mehreren Maschinen zu verteilen. Die Abfrage wird parallel ausgeführt. Einige DB2 UDB-Dienstprogramme arbeiten ebenfalls mit dieser Art der Parallelität.

Bei der partitionsübergreifenden Parallelität wird das, was im Allgemeinen als eine einzige Datenbankoperation betrachtet wird, (z. B. eine Indexerstellung, das Laden von Daten, SQL-Abfragen) in mehrere Teile unterteilt, von denen viele oder alle parallel *über mehrere Partitionen einer partitionierten Datenbank hinweg auf einer oder mehreren Maschinen* ausgeführt werden können.

Abb. 14 zeigt eine Abfrage, die in vier Teile aufgeteilt ist, die parallel ausgeführt werden können, wobei die Ergebnisse schneller zurückgeliefert werden als bei serieller Ausführung in einer Einzelpartition.

Der Grad der Parallelität wird im Wesentlichen durch die Anzahl der Partitionen, die Sie erstellen, und die Art und Weise der Definition der Partitionsgruppen bestimmt.

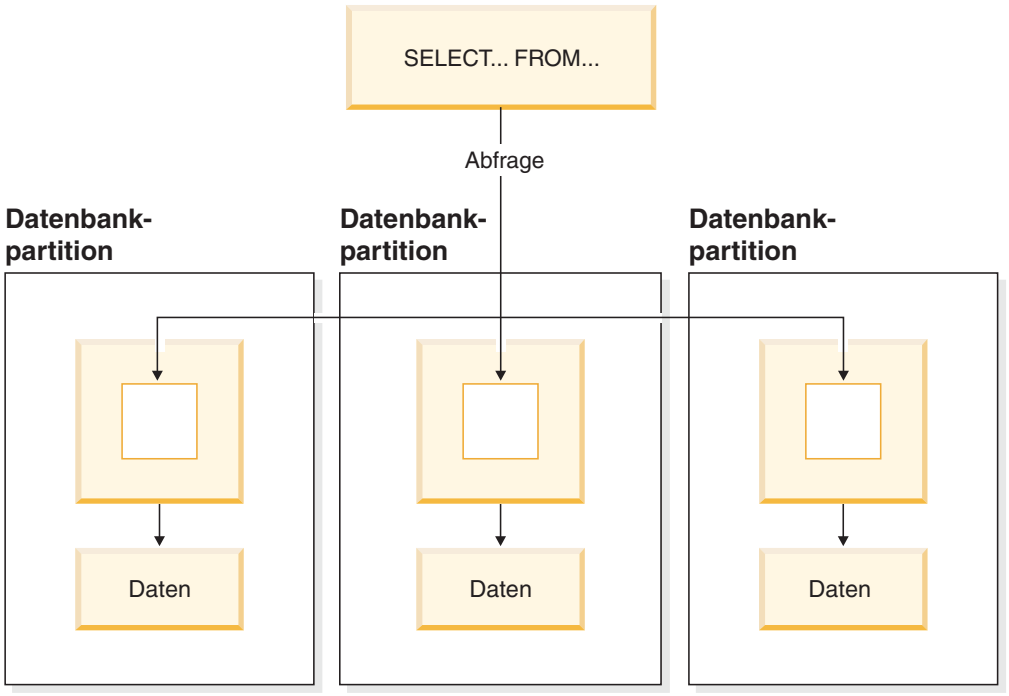


Abbildung 14. Partitionsübergreifende Parallelität

Simultane partitionsinterne und partitionsübergreifende Parallelität

Sie können die partitionsinterne Parallelität und die partitionsübergreifende Parallelität gleichzeitig nutzen. Diese Kombination bietet zwei Dimensionen der Parallelität, wodurch sich ein weiterer wesentlicher Anstieg der Verarbeitungsgeschwindigkeit für Abfragen ergibt.

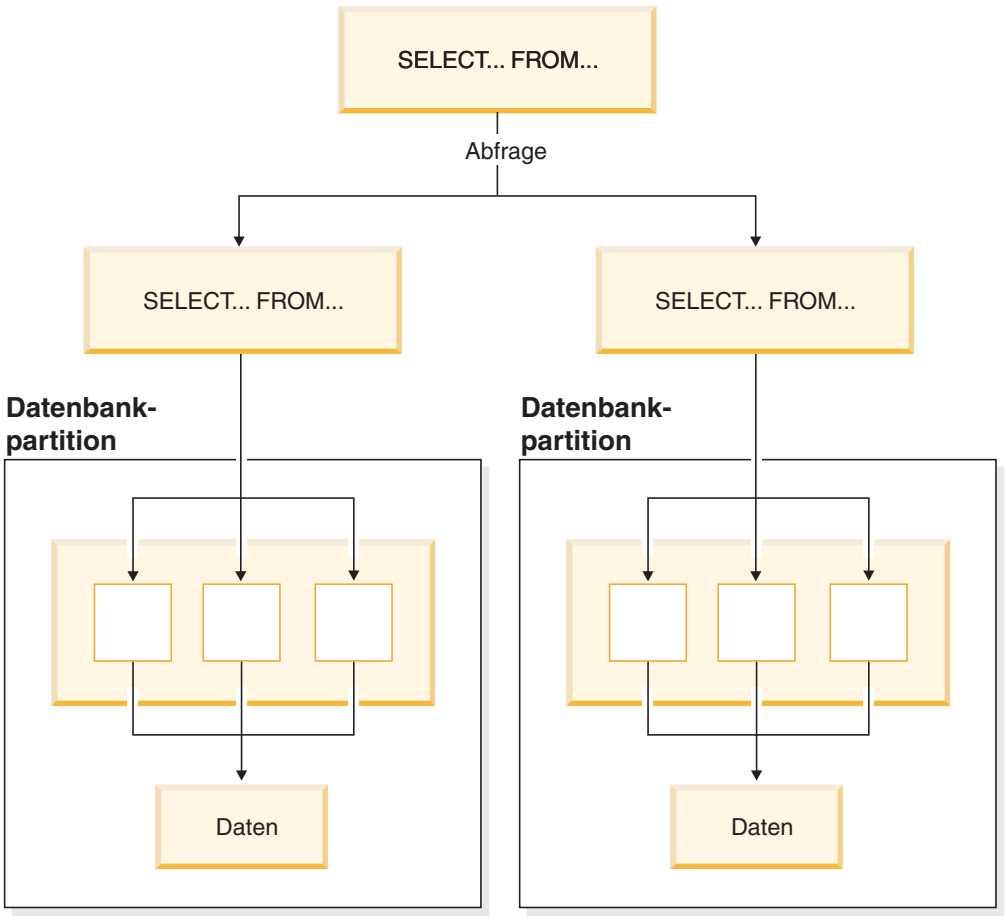


Abbildung 15. Simultane partitionsinterne und partitionsübergreifende Parallelität

Dienstprogrammparallelität

DB2 UDB-Dienstprogramme können mit partitionsinterner Parallelität arbeiten. Sie können außerdem die Vorteile der partitionsübergreifenden Parallelität nutzen. Wenn mehrere Datenbankpartitionen vorhanden sind, werden die Dienstprogramme in jeder der Partitionen parallel ausgeführt.

Das Dienstprogramm LOAD kann die Vorteile der partitionsinternen Parallelität und der E/A-Parallelität nutzen. Das Laden von Daten in eine Tabelle ist eine CPU-intensive Operation. Das Dienstprogramm LOAD nutzt die Möglichkeit mehrerer Prozessoren bei Operationen wie dem Analysieren und Formatieren von Daten. Darüber hinaus kann es parallele E/A-Server zum gleichzeitigen Schreiben der Daten in Behälter verwenden.

In einer Umgebung mit partitionierten Datenbanken nutzt das Dienstprogramm LOAD die Vorteile der partitionsinternen, partitionsübergreifenden und der E/A-Parallelität durch parallele Aufrufe in allen Datenbankpartitionen, in denen sich die Tabelle befindet.

Während der Indexerstellung erfolgt das Durchsuchen und das nachfolgende Sortieren der Daten parallel. DB2 UDB nutzt sowohl die E/A-Parallelität als auch die partitionsinterne Parallelität bei der Erstellung eines Index. Dadurch wird die Indexerstellung bei der Verarbeitung der Anweisung CREATE INDEX, beim Neustart (wenn ein Index als ungültig markiert ist) und bei der Reorganisation von Daten beschleunigt.

Sichern und Wiederherstellen von Daten sind Operationen, die wesentlich von der E/A-Leistung abhängig sind. DB2 UDB nutzt sowohl die E/A-Parallelität als auch partitionsinterne Parallelität bei der Durchführung von BACKUP- und RESTORE-Operationen. Der Befehl BACKUP nutzt die E/A-Parallelität, indem er von mehreren Tabellenbereichsbehältern parallel liest und asynchron auf mehrere Sicherungsmedien parallel schreibt.

Zugehörige Konzepte:

• „Partitions- und Prozessorumgebungen" auf Seite 38

Partitions- und Prozessorumgebungen

Dieser Abschnitt enthält eine Übersicht über folgende Hardwareumgebungen:

- Einzelpartition auf Einzelprozessormaschine
- Einzelpartition auf Mehrprozessormaschine (SMP)
- Konfigurationen mit mehreren Partitionen
  - Partitionen mit einem Prozessor (MPP)
  - Partitionen mit mehreren Prozessoren (Cluster aus SMP-Systemen)
  - Logische Datenbankpartitionen (in DB2® Parallel Edition für AIX® Version 1 auch als MLN - Multiple Logical Nodes bezeichnet)

Für jede Umgebung werden die Kapazität und Skalierbarkeit behandelt. *Kapazität* bezieht sich hier auf die Anzahl der Benutzer und Anwendungen, die auf die Datenbank zugreifen können. Dies wird großenteils durch Hauptspeicher, Agenten, Sperren, E/A-Operationen und Speicherverwaltung bestimmt. *Skalierbarkeit* bezeichnet die Fähigkeit einer Datenbank, bei zunehmender Größe weiterhin die gleichen Betriebsmerkmale und Antwortzeiten zu zeigen.

Einzelpartition auf einer Einzelprozessormaschine

Diese Umgebung verfügt über Hauptspeicher und Platte, enthält aber nur eine CPU (siehe Abb. 16 auf Seite 39). Für diese Umgebung werden viele verschiedene Bezeichnungen verwendet, wie zum Beispiel eigenständige Datenbank, Client-/Serverdatenbank, serielle Datenbank, Einprozessorsystem oder nicht parallele bzw. Einzelknotenumgebung.

Die Datenbank in dieser Umgebung erfüllt die Anforderungen einer Abteilung oder eines kleinen Büros, wobei die Daten und Systemressourcen (einschließlich des Einzelprozessors bzw. einer CPU) von einem einzelnen Datenbankmanager verwaltet werden.

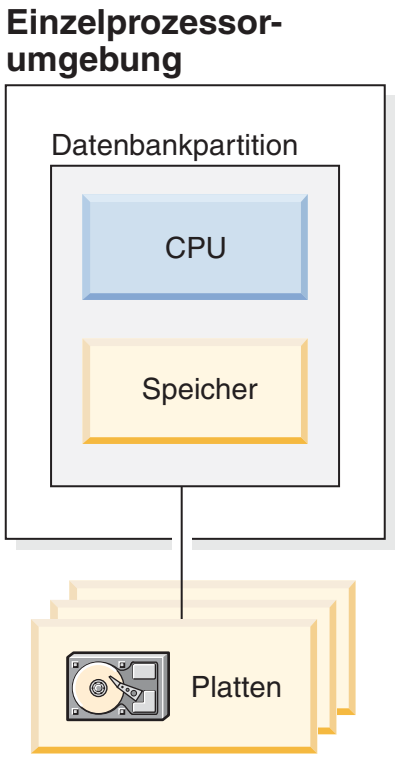


Abbildung 16. Einzelpartition auf Einzelprozessormaschine

Kapazität und Skalierbarkeit

In dieser Umgebung können Sie weitere Platten hinzufügen. Durch den Einsatz eines oder mehrerer E/A-Server für jede Platte kann mehr als eine E/A-Operation gleichzeitig stattfinden.

Ein Einzelprozessorsystem ist durch die Menge an Plattenspeicherplatz, den der Prozessor handhaben kann, eingeschränkt. Ungeachtet aller zusätzlichen Komponenten wie Arbeitsspeicher oder Festplattenspeicher, die Sie zur schnelleren Verarbeitung von Benutzeranforderungen vielleicht hinzufügen, kann eine einzelne CPU bei weiterer Zunahme der Auslastung Benutzeranforderungen nicht mehr schneller verarbeiten. Wenn Sie das Maximum an Kapazität und Skalierbarkeit erreicht haben, können Sie die Umrüstung auf ein Einzelpartitionssystem mit mehreren Prozessoren in Erwägung ziehen.

Einzelpartition auf Mehrprozessormaschine

Diese Umgebung besteht gewöhnlich aus mehreren gleich starken Prozessoren innerhalb derselben Maschine (siehe Abb. 17 auf Seite 40) und wird als *symmetrisches Mehrprozessorsystem (SMP-System)* bezeichnet (SMP - Symmetric Multiprocessor). Ressourcen wie Plattenspeicherplatz und Hauptspeicher werden *gemeinsam* benutzt.

Wenn mehrere Prozessoren verfügbar sind, können verschiedene Datenbankoperationen schneller durchgeführt werden. DB2 Universal Database™ (DB2 UDB) kann auch die Arbeit einer einzigen Abfrage auf die verfügbaren Prozessoren verteilen, um die Verarbeitungsgeschwindigkeit zu erhöhen.

Andere Datenbankoperationen wie das Laden von Daten, das Sichern und Wiederherstellen von Tabellenbereichen sowie die Erstellung von Indizes auf vorhandenen Daten können die Verfügbarkeit mehrerer Prozessoren ausnutzen.

Symmetrische Multiprozessorumgebung (SMP)

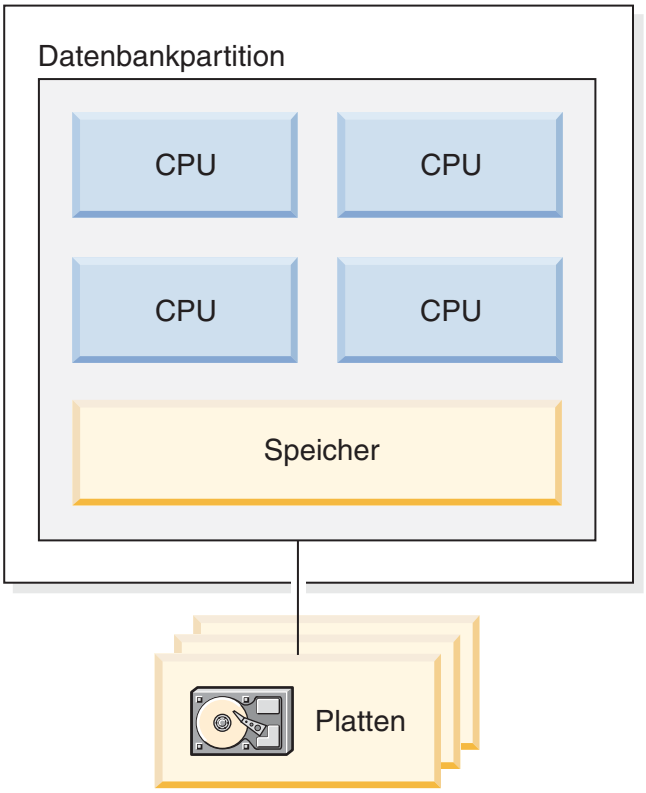


Abbildung 17. Einzelpartitionsdatenbank in einer symmetrischen Mehrprozessorumgebung

Kapazität und Skalierbarkeit

In dieser Umgebung können Sie weitere Prozessoren hinzufügen. Da aber verschiedene Prozessoren versuchen könnten, gleichzeitig auf dieselben Daten zuzugreifen, können mit der Zunahme der Geschäftsoperationen Einschränkungen auftreten. Durch die gemeinsame Benutzung von Hauptspeicher und Platten werden effektiv sämtliche Datenbankdaten gemeinsam benutzt.

Sie können die E/A-Kapazität der Datenbankpartition, die Ihrem Prozessor zugeordnet ist, durch eine größere Anzahl von Platten erhöhen. Sie können E/A-Server speziell zur Verarbeitung von E/A-Anforderungen einrichten. Durch den Einsatz eines oder mehrerer E/A-Server für jede Platte kann mehr als eine E/A-Operation gleichzeitig stattfinden.

Wenn Sie das Maximum an Kapazität und Skalierbarkeit erreicht haben, können Sie die Umrüstung auf ein System mit mehreren Partitionen in Erwägung ziehen.

Konfigurationen mehrerer Partitionen

Sie können eine Datenbank in mehrere Partitionen aufteilen, die sich jeweils auf einer eigenen Maschine befinden. Mehrere Maschinen mit mehreren Datenbankpartitionen können in einer Gruppe zusammengefasst werden. In diesem Abschnitt werden die folgenden Partitionskonfigurationen beschrieben:

- Partitionen auf Systemen mit einem Prozessor
- Partitionen auf Systemen mit mehreren Prozessoren
- Logische Datenbankpartitionen

Partitionen mit einem Prozessor

In dieser Umgebung gibt es viele Datenbankpartitionen. Jede Partition befindet sich auf einer eigenen Maschine mit eigenem Prozessor, eigenem Hauptspeicher und eigenen Platten (Abb. 18). Alle Maschinen sind über eine Kommunikationseinrichtung verbunden. Für eine solche Umgebung werden viele verschiedene Bezeichnungen gebraucht, wie zum Beispiel: Cluster, Cluster von Einzelprozessoren, Umgebung mit exklusiver Parallelverarbeitung (MPP - Massively Parallel Processing) oder Konfiguration ohne gemeinsame Nutzung von Ressourcen. Die letztgenannte Bezeichnung charakterisiert die Anordnung der Ressourcen. Im Gegensatz zu einer SMP-Umgebung findet in einer MPP-Umgebung keine gemeinsame Nutzung von Hauptspeicher oder Platten statt. In der MPP-Umgebung fallen daher auch die Einschränkungen aufgrund der gemeinsamen Nutzung von Hauptspeicher und Platten fort.

Durch die Partitionierung in einer Umgebung kann eine Datenbank trotz ihrer physischen Verteilung auf mehrere Partitionen eine logische Gesamtheit bilden. Die Tatsache, dass die Daten auf Partitionen verteilt sind, bleibt für die Mehrheit der Benutzer transparent. Die Arbeit kann unter den Datenbankmanagern aufgeteilt werden. Die Datenbankmanager in den einzelnen Partitionen verrichten die Arbeit jeweils am eigenen Teil der Datenbank.

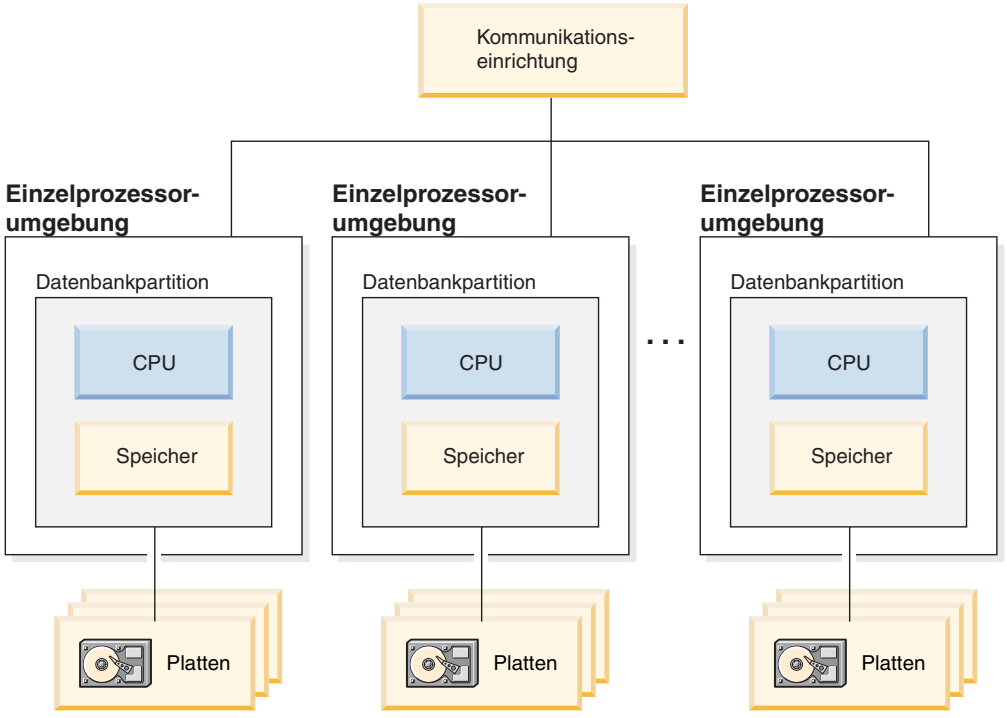


Abbildung 18. Umgebung zur exklusiven Parallelverarbeitung (MPP-Umgebung)

Kapazität und Skalierbarkeit: In dieser Umgebung können Sie weitere Datenbankpartitionen (Knoten) Ihrer Konfiguration hinzufügen. Auf einigen Plattformen, z. B. RS/6000® SP™, ist die mögliche maximale Anzahl 512 Knoten. Jedoch kann es praktische Grenzen für die Verwaltung einer höheren Anzahl von Maschinen und Exemplaren geben.

Wenn Sie das Maximum an Kapazität und Skalierbarkeit erreicht haben, können Sie die Umrüstung auf ein System in Erwägung ziehen, in dem jede Partition mehrere Prozessoren hat.

Partitionen mit mehreren Prozessoren

Eine Alternative zu einer Konfiguration, in der jede Partition einen einzigen Prozessor hat, ist eine Konfiguration, in der eine Partition über mehrere Prozessoren verfügt. Eine solche Konfiguration wird als *SMP-Cluster* bezeichnet (Abb. 19).

Diese Art der Konfiguration verbindet die Vorteile von SMP- und MPP-Parallelität. Dies bedeutet, dass eine Abfrage in einer Einzelpartition mit Hilfe mehrerer Prozessoren durchgeführt werden kann. Darüber hinaus kann eine Abfrage auch parallel in mehreren Partitionen durchgeführt werden.

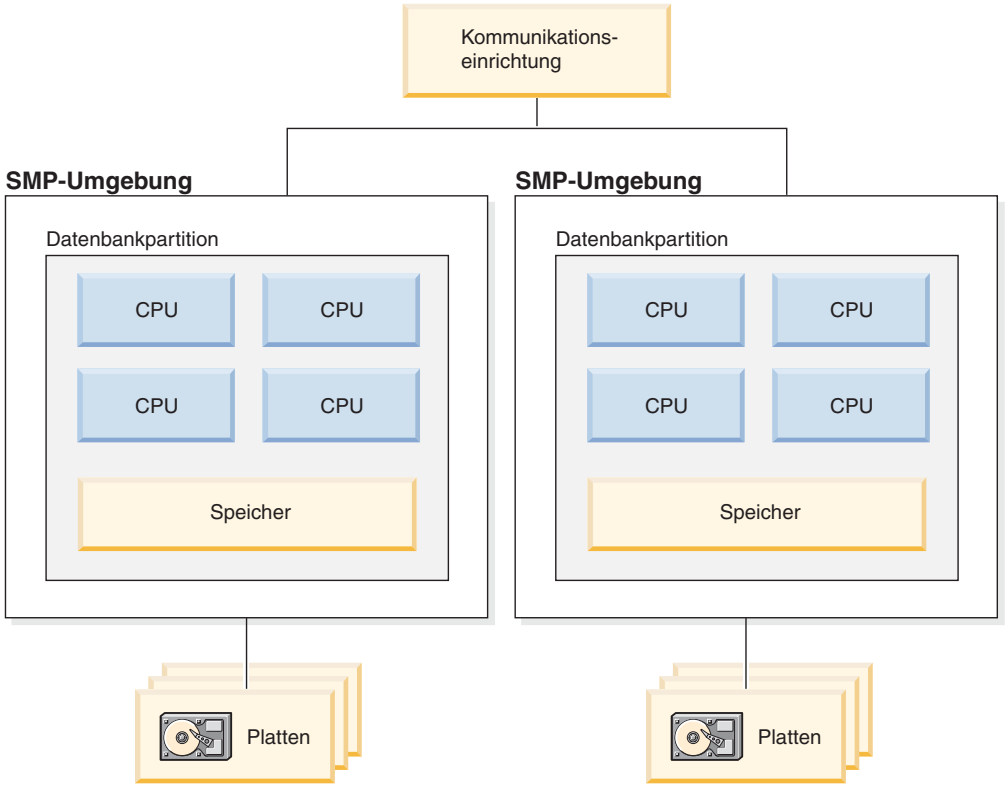


Abbildung 19. Mehrere symmetrische Mehrprozessorumgebungen (SMP) in einem Cluster

Kapazität und Skalierbarkeit: In dieser Umgebung können Sie weitere Datenbankpartitionen und den vorhandenen Datenbankpartitionen weitere Prozessoren hinzufügen.

Logische Datenbankpartitionen

Eine logische Datenbankpartition unterscheidet sich von einer physischen Partition darin, dass ihr nicht die Steuerung einer ganzen Maschine unterliegt. Obwohl die Maschine über gemeinsam benutzte Ressourcen verfügt, nutzen Datenbankpartitionen die Ressourcen nicht gemeinsam. Prozessoren werden gemeinsam benutzt, Platten und Hauptspeicher jedoch nicht.

Logische Datenbankpartitionen bieten Skalierbarkeit. Mehrere Datenbankmanager, die in mehreren logischen Partitionen ausgeführt werden, können die verfügbaren Ressourcen eventuell erschöpfender nutzen, als dies ein einzelner Datenbankmanager könnte. Abb. 20 veranschaulicht die Möglichkeit, mehr Skalierbarkeit auf einer SMP-Maschine zu gewinnen, indem weitere Partitionen hinzugefügt werden. Dies gilt insbesondere für Maschinen mit vielen Prozessoren. Durch die Partitionierung der Datenbank können Sie jede Partition getrennt verwalten und wiederherstellen.

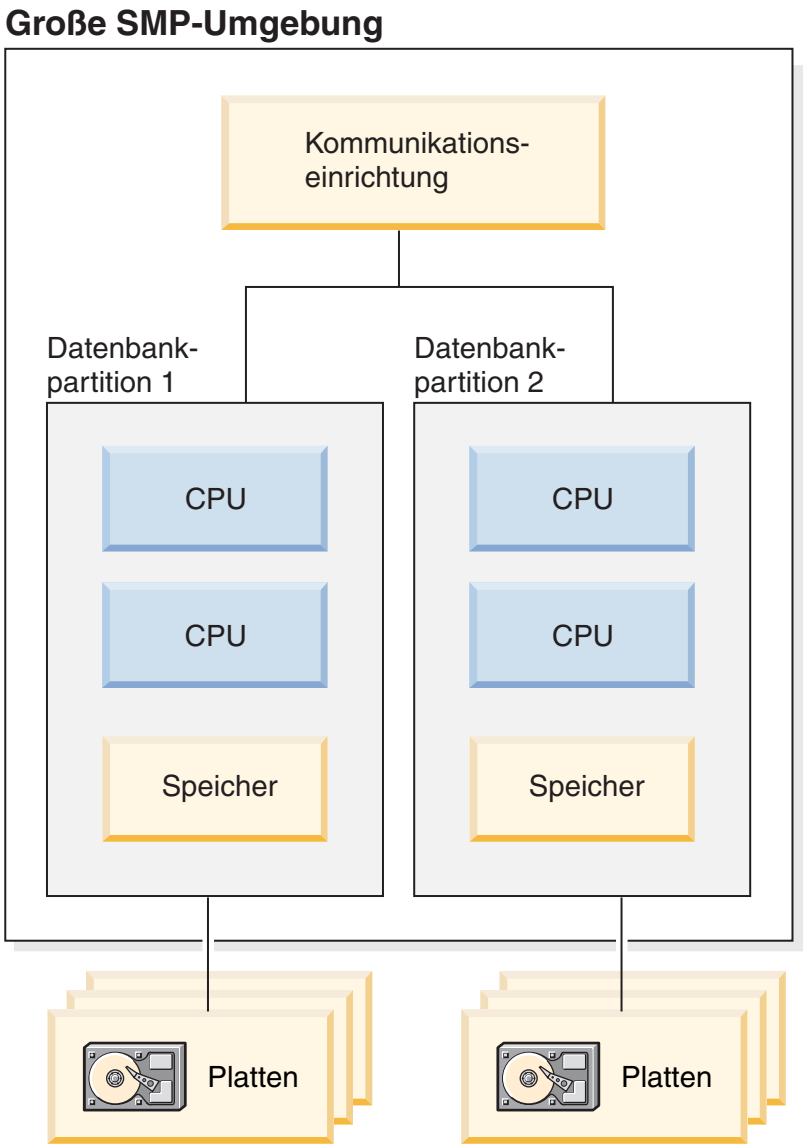


Abbildung 20. Partitionierte Datenbank mit symmetrischer Mehrprozessorumgebung

Abb. 21 veranschaulicht die Möglichkeit, die in Abb. 20 auf Seite 43 gezeigte Konfiguration zu vervielfachen, um die Verarbeitungsleistung zu erhöhen.

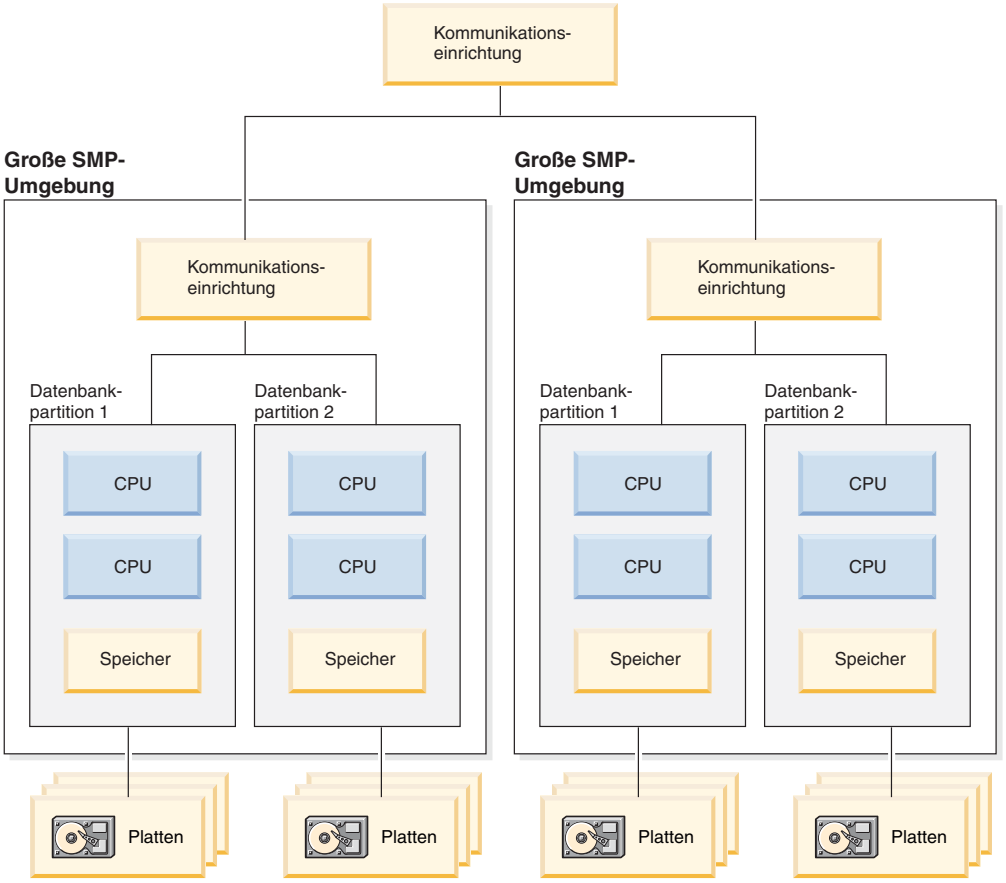


Abbildung 21. Partitionierte Datenbank mit SMP-Umgebungen in einem Cluster

Anmerkung: Die Möglichkeit, zwei oder mehr Partitionen nebeneinander auf derselben Maschine zu haben (ungeachtet der Anzahl der Prozessoren), bietet eine größere Flexibilität bei der Einrichtung hochverfügbarer Konfigurationen und der Implementierung von Übernahmestrategien bei Systemausfällen. Bei Ausfall einer Maschine kann eine Datenbankpartition automatisch von einer anderen Maschine, die bereits eine andere Partition derselben Datenbank enthält, übernommen und wieder gestartet werden.

Zusammenfassung der geeigneten Parallelität für die einzelnen Hardwareumgebungen

Die folgende Tabelle gibt eine Übersicht über die Arten der Parallelität, mit denen die verschiedenen Hardwareumgebungen am besten genutzt werden können.

Tabelle 3. Mögliche Arten der Parallelität in jeder Hardwareumgebung

Hardwareumgebung	**E/A-Parallelität**	**Abfrageinterne Parallelität**	
		Partitionsinterne Parallelität	**Partitionsübergreifende Parallelität**
Einzelpartition, Einzelprozessor	Ja	Nein (1)	Nein
Einzelpartition, mehrere Prozessoren (SMP)	Ja	Ja	Nein
Mehrere Partitionen, ein Prozessor (MPP)	Ja	Nein (1)	Ja
Mehrere Partitionen, mehrere Prozessoren (Cluster von SMP-System)	Ja	Ja	Ja
Logische Datenbankpartitionen	Ja	Ja	Ja
Anmerkung: (1) Es kann vorteilhaft sein, den Grad der Parallelität (mit Hilfe eines der Konfigurationsparameter) auch auf einem Einzelprozessorsystem auf einen Wert größer als 1 zu setzen, besonders wenn die von Ihnen ausgeführten Abfragen die CPU nicht voll auslasten (z. B. wenn die Abfragen E/A-lastig sind).			

Zugehörige Konzepte:

• „Parallelität" auf Seite 34

Kapitel 3. Data Warehouses

Welche Lösungen bietet Data Warehousing?

Systeme mit *Betriebsdaten* (die Daten, mit denen Ihre täglichen Geschäftstransaktionen ausgeführt werden) enthalten Informationen, die für Geschäftsanalytiker hilfreich sind. Beispielsweise können Analytiker mit Hilfe der Angaben, welche Produkte in welchen Regionen zu welcher Jahreszeit verkauft wurden, nach Unregelmäßigkeiten suchen oder künftige Verkaufszahlen prognostizieren.

Es können jedoch diverse Probleme auftreten, wenn die Analytiker direkt auf die Betriebsdaten zugreifen:

- Analytiker haben möglicherweise nicht die notwendige Erfahrung, um die Betriebsdatenbank abzufragen. Die Abfrage von IMS™-Datenbanken beispielsweise setzt ein Anwendungsprogramm voraus, das einen speziellen Typ der Datenbearbeitungssprache verwendet. Im Allgemeinen sind Programmierer, die das Wissen und die Erfahrung zur Abfrage der Betriebsdatenbank haben, voll mit der Pflege der Datenbank und ihrer Anwendungen ausgelastet.
- Bei vielen Betriebsdatenbanken, z. B. Datenbanken für eine Bank, ist die Leistung das Entscheidende. Das System kann Sofortabfragen von Benutzern nicht handhaben.
- Die Betriebsdaten liegen im Allgemeinen nicht im besten Format zur Verwendung für Geschäftsanalysen vor. Verkaufsdaten, die nach Produkt, Region und Jahreszeit zusammengefasst sind, sind für Analytiker viel sinnvoller als die Rohdaten.

Data Warehousing löst diese Probleme. Beim *Data Warehousing* können Sie Sammlungen von *Informationsdaten* erstellen. Informationsdaten sind Daten, die aus den Betriebsdaten extrahiert und dann zur Entscheidungsfindung umgesetzt werden. Beispielsweise kopiert ein Data Warehousing-Tool alle Verkaufsdaten aus der Betriebsdatenbank, bereinigt die Daten, führt Berechnungen durch, um die Daten zusammenzufassen, und schreibt die zusammengefassten Daten in eine separate Datenbank. Die Benutzer können diese separate Datenbank (das *Warehouse*) ohne Auswirkungen auf die Betriebsdatenbanken abfragen.

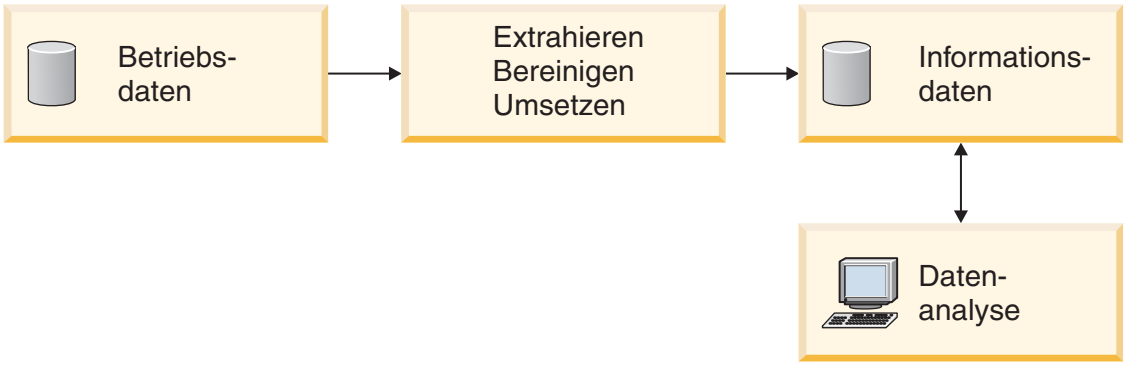


Abbildung 22. Der Pfad von den Betriebsdaten zur Datenanalyse

Zugehörige Konzepte:

- „Data Warehouse-Objekte" auf Seite 48

Data Warehouse-Objekte

In den folgenden Abschnitten werden die Objekte beschrieben, mit denen Sie Ihr Data Warehouse erstellen und verwalten.

Themenbereiche

Ein Themenbereich bezeichnet und gruppiert die Prozesse, die zu einem logischen Geschäftsbereich gehören. Wenn Sie z. B. ein Warehouse mit Marketing- und Verkaufsdaten erstellen, definieren Sie einen Themenbereich "Verkauf" und einen Themenbereich "Marketing". Danach fügen Sie die verkaufsbezogenen Prozesse dem Themenbereich "Verkauf" hinzu. Definitionen, die sich auf die Marketingdaten beziehen, fügen Sie analog dazu dem Themenbereich "Marketing" hinzu.

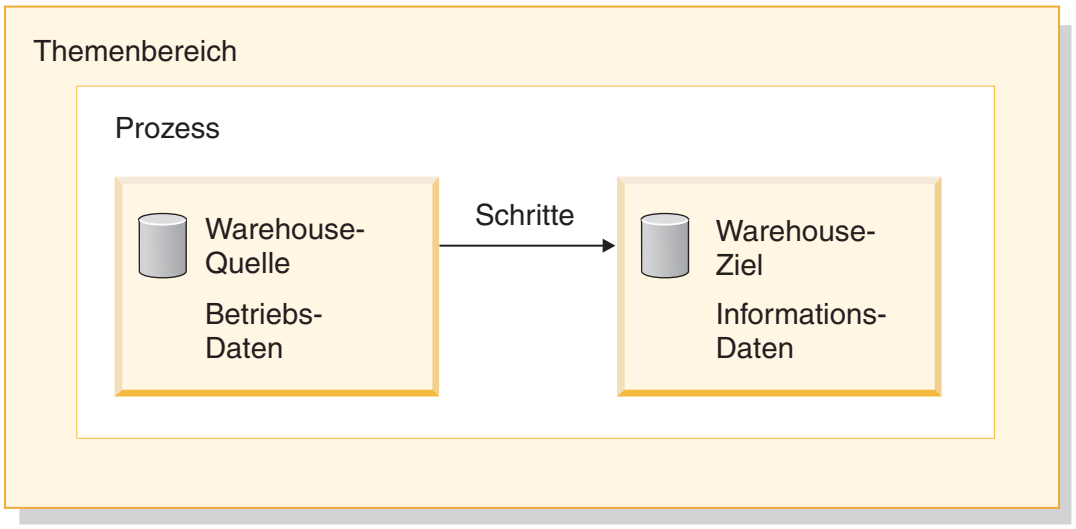


Abbildung 23. Die Hierarchie von Themenbereichen

Warehouse-Quellen

Warehouse-Quellen geben die Tabellen und Dateien an, die die Daten für Ihr Warehouse liefern. Die Data Warehouse-Zentrale verwendet die Spezifikationen in den Warehouse-Quellen für den Zugriff auf die Daten. Quelle kann fast jede beliebige relationale oder nicht relationale Quelle (Tabelle, Sicht, Datei) oder WebSphere® Site Analyzer-Quelle sein, die mit Ihrem Netzwerk verbunden werden kann.

Warehouse-Ziele

Warehouse-Ziele sind Datenbanktabellen oder -dateien, die umgesetzte Daten enthalten. Warehouse-Ziele können selbst wiederum dazu verwendet werden, Daten für andere Warehouse-Ziele bereitzustellen. Ein zentrales Warehouse kann zum Beispiel Daten für Abteilungsserver, eine Hauptfakttabelle im Warehouse kann Daten für Übersichtstabellen bereitstellen.

Warehouse-Steuerungsdatenbanken

Die Warehouse-Steuerungsdatenbank enthält die Steuertabellen, die zum Speichern der Metadaten der Data Warehouse-Zentrale erforderlich sind. Mit der Version 8.2 der Data Warehouse-Zentrale muss die Warehouse-Steuerungsdatenbank eine Datenbank im UTF-8-Format (Unicode Transformation Format bzw. Unicode) sein. Diese Voraussetzung ermöglicht eine erweiterte Sprachenunterstützung für die Data Warehouse-Zentrale. Wenn Sie versuchen, sich an der Data Warehouse-Zen-

trale unter Verwendung einer Datenbank anzumelden, die nicht im Unicode-Format ist, empfangen Sie eine Fehlernachricht, dass Sie sich nicht anmelden können. Sie können das Verwaltungstool für die Warehouse-Steuerungsdatenbank (Warehouse Control Database Management Tool) zur Migration der Metadaten aus einer angegebenen Datenbank in eine neue Unicode-Datenbank verwenden.

Warehouse-Agenten und Agentensites

Warehouse-Agenten verwalten den Datenfluss zwischen den Datenquellen und den Ziel-Warehouses. Warehouse-Agenten sind verfügbar für die Betriebssysteme AIX®, Linux, iSeries™, z/OS™, Windows® NT, Windows 2000 und Windows XP sowie für Solaris™ Operating Environment. Die Agenten verwenden ODBC-Treiber (Open Database Connectivity) oder DB2® CLI zur Kommunikation mit unterschiedlichen Datenbanken.

Mehrere Agenten können die Übertragung von Daten zwischen Quellen und Ziel-Warehouses handhaben. Die Anzahl von Agenten, die Sie verwenden, hängt von Ihrer bestehenden Konnektivitätskonfiguration und der Datenmenge ab, die Sie in Ihr Warehouse versetzen wollen. Weitere Exemplare eines Agenten können erstellt werden, wenn mehrere Prozesse, für die derselbe Agent erforderlich ist, gleichzeitig ausgeführt werden.

Agenten können lokale oder ferne Agenten sein. Ein lokaler Warehouse-Agent ist ein Agent, der auf derselben Workstation installiert ist wie der Warehouse-Server. Ein ferner Warehouse-Agent ist ein Agent, der auf einer anderen Workstation mit Konnektivität zum Warehouse-Server installiert ist.

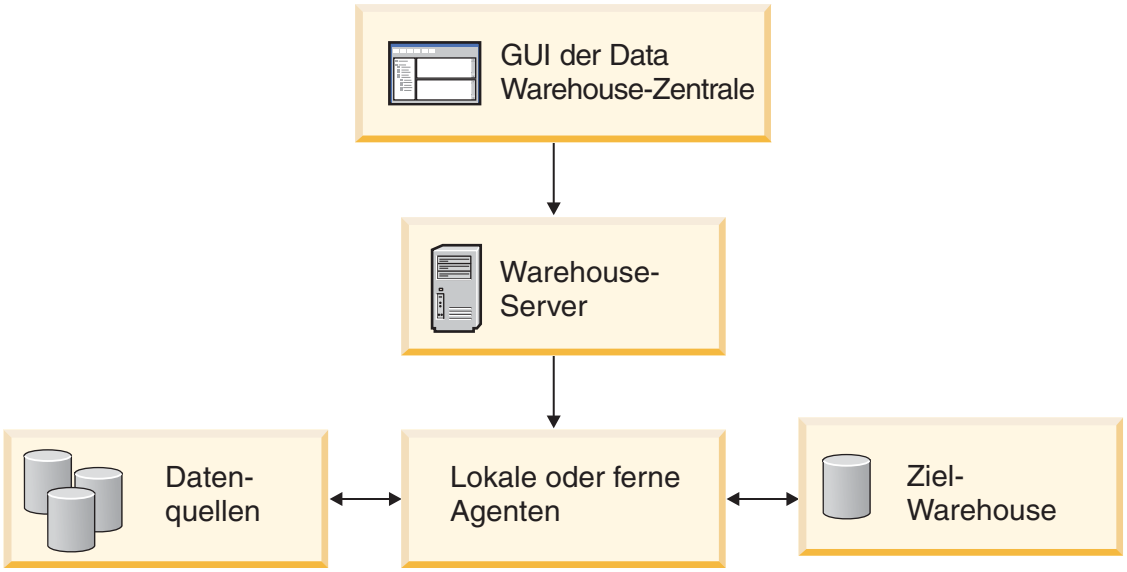


Abbildung 24. Die Beziehung zwischen Agenten, Datenquellen, Ziel-Warehouses und dem Warehouse-Server

Eine Agentensite ist ein logischer Name für eine Workstation, auf der Agentensoftware installiert ist. Der Name der Agentensite ist nicht mit dem TCP/IP-Hostnamen identisch. Einer einzelnen Workstation kann nur ein TCP/IP-Hostname zugeordnet sein. Es können jedoch mehrere Agentensites auf einer Workstation definiert werden. Jede Agentensite wird durch einen logischen Namen angegeben.

Die Standardagentensite, die so genannte Standard-DWZ-Agentensite, ist ein lokaler Agent, den die Data Warehouse-Zentrale während der Initialisierung der Warehouse-Steuerungsdatenbank definiert.

Prozesse und Schritte

Ein Prozess enthält eine Reihe von Schritten, die eine Umsetzung und eine Versetzung von Daten für eine bestimmte Warehouse-Verwendung durchführen. Im Allgemeinen versetzt ein Prozess Quellendaten in das Warehouse. Die Daten werden dann für die Warehouse-Verwendung zusammengefasst. Ein Prozess kann eine einzige unstrukturierte Tabelle oder eine Gruppe von Übersichtstabellen erstellen. Ein Prozess führt möglicherweise auch eine spezielle Art der Datenumsetzung aus.

Ein *Schritt* ist die Definition einer einzelnen Operation im Warehouse. Durch die Verwendung von SQL-Anweisungen und aufrufenden Programmen definieren Schritte, wie Daten versetzt und umgesetzt werden sollen. Beim Ausführen eines Schritts kann eine Datenübertragung zwischen der Warehouse-Quelle und dem Warehouse-Ziel oder eine beliebige Umsetzung dieser Daten stattfinden.

Ein Schritt ist eine logische Einheit in der Data Warehouse-Zentrale, die Folgendes definiert:
- Programmverbindung (Link) zu den Quellendaten
- Definition der Ausgabetabelle oder -datei und entsprechende Programmverbindung
- Mechanismus (eine SQL-Anweisung oder ein Programm) und Definition zum Füllen der Ausgabetabelle oder -datei
- Verarbeitungsoptionen und Zeitplan für das Füllen der Ausgabetabelle oder -datei

Angenommen, die Data Warehouse-Zentrale soll die folgenden Tasks ausführen:
1. Extrahieren von Daten aus verschiedenen Datenbanken
2. Umsetzen der Daten in ein Format
3. Schreiben der Daten in eine Tabelle in einem Daten-Warehouse

Sie würden dann einen Prozess erstellen, der mehrere Schritte enthält. Jeder Schritt führt eine eigene Task durch, z. B. Extrahieren der Daten aus der Datenbank oder Konvertieren der Daten in das korrekte Format. Möglicherweise müssen Sie mehrere Schritte erstellen, um die Daten vollständig umzusetzen und zu formatieren und sie in die Zieltabelle aufzunehmen.

Die Ausführung eines Schritts oder Prozesses kann sich folgendermaßen auf das Ziel auswirken:
- Ersetzen aller Daten im Warehouse-Ziel durch neue Daten
- Anhängen der neuen Daten an die bereits vorhandenen Daten
- Anhängen einer separaten Edition von Daten
- Aktualisieren vorhandener Daten

Sie können einen Schritt entweder bei Bedarf ausführen oder den Schritt zur Ausführung wie folgt terminieren:
- Zu einem festgelegten Zeitpunkt
- Nur einmal
- Wiederholt, zum Beispiel jeden Freitag
- Sequenziell, so dass nach Abschluss des einen Schritts die Ausführung des nächsten Schritts beginnt
- Nach dem erfolgreichen oder nicht erfolgreichen Abschluss eines anderen Schritts

Wenn Sie einen Zeitplan für einen Prozess definieren, wird der erste Schritt im Prozess zum definierten Zeitpunkt ausgeführt.

In den folgenden Abschnitten werden die unterschiedlichen Arten von Schritten beschrieben, die Sie in der Data Warehouse-Zentrale finden.

SQL-Schritte

Die Data Warehouse-Zentrale stellt zwei Typen von SQL-Schritten zur Verfügung. Der Schritt **SQL SELECT und INSERT** verwendet eine SQL-Anweisung SELECT, um Daten aus einer Warehouse-Quelle zu extrahieren, und generiert eine Anweisung INSERT, um die Daten in die Warehouse-Zieltabelle einzufügen. Der Schritt **SQL SELECT und UPDATE** verwendet eine SQL-Anweisung SELECT, um Daten aus einer Warehouse-Quelle zu extrahieren und die in der Warehouse-Zieltabelle vorhandenen Daten zu aktualisieren.

Programmschritte

Die Data Warehouse-Zentrale stellt verschiedene Typen von Programmschritten zur Verfügung: DB2 für iSeries-Programme, DB2 für z/OS-Programme, DB2 Universal Database™-Programme, Visual Warehouse™ 5.2 DB2-Programs, OLAP Server-Programme, Dateiprogramme und Replikationsprogramme. Diese Schritte führen vordefinierte Programme und Dienstprogramme aus. Die Warehouse-Programme für ein bestimmtes Betriebssystem sind mit dem Agenten für dieses Betriebssystem in einem Paket zusammengefasst. Die Warehouse-Programme werden beim Installieren des Agentencodes installiert.

Umsetzungsschritte

Umsetzungsschritte sind gespeicherte Prozeduren und benutzerdefinierte Funktionen, die Statistik- oder Warehouse-Umsetzungsprogramme angeben, mit denen Sie Daten umsetzen können. Umsetzungsprogramme können zum Bereinigen, Vertauschen und Umlagern von Daten, zum Generieren von Primärschlüsseln und Periodentabellen und zum Berechnen verschiedener Statistiken verwendet werden.

In einem Umsetzungsschritt geben Sie eines der Statistik- oder Warehouse-Umsetzungsprogramme an. Wenn Sie den Prozess ausführen, schreibt der Umsetzungsschritt Daten in eines oder mehrere Warehouse-Ziele.

Benutzerdefinierte Programmschritte

Ein benutzerdefinierter Programmschritt ist eine logische Einheit in der Data Warehouse-Zentrale, die eine geschäftsspezifische Umsetzung darstellt, die von der Data Warehouse-Zentrale gestartet werden soll. Da jedes Unternehmen individuelle Anforderungen für die Datenumsetzung hat, können Unternehmen ihre eigenen Programmschritte schreiben oder Tools einsetzen, die von anderen Anbietern, z. B. von ETI oder Vality, bereitgestellt werden.

Sie können beispielsweise ein benutzerdefiniertes Programm schreiben, das die folgenden Funktionen ausführt:

1. Exportieren von Daten aus einer Tabelle
2. Bearbeiten der Daten
3. Schreiben der Daten in eine temporäre Ausgaberessource oder ein Warehouse-Ziel

Zugehörige Konzepte:

- „Welche Lösungen bietet Data Warehousing?" auf Seite 47
- „Warehouse-Tasks" auf Seite 52
- „Was ist ein benutzerdefiniertes Programm?" in *Data Warehouse-Zentrale Verwaltung*
- „Was ist eine Programmgruppe?" in *Data Warehouse-Zentrale Verwaltung*

Warehouse-Tasks

Die Erstellung eines Data Warehouses umfasst die folgenden Tasks:

- Angeben der Quellendaten (oder Betriebsdaten) und Definieren dieser Daten zur Verwendung als Warehouse-Quellen
- Erstellen einer Datenbank, die als Warehouse verwendet werden soll, sowie Definieren von Warehouse-Zielen
- Definieren eines Themenbereichs für Gruppen von Prozessen, die Sie in Ihrem Warehouse definieren
- Angeben der Art und Weise, wie die Quellendaten versetzt und in das entsprechende Format für die Warehouse-Datenbank umgesetzt werden sollen, durch das Definieren von Schritten in den Prozessen
- Testen der von Ihnen definierten Schritte und Aufstellen eines Zeitplans für die automatische Ausführung
- Verwalten des Warehouses durch Definieren der Sicherheit und Überwachen der Datenbanknutzung mit Hilfe des Notizbuchs **Laufende Prozesse**.

Wenn Sie über DB2® Warehouse Manager verfügen, können Sie einen Informationskatalog der Daten im Warehouse erstellen. Ein Informationskatalog ist eine Datenbank, die Geschäftsmetadaten enthält. Geschäftsmetadaten helfen Benutzern, die für sie in der Organisation verfügbaren Daten und Informationen anzugeben und zu lokalisieren. Data Warehouse-Metadaten können im Informationskatalog veröffentlicht werden. Der Informationskatalog kann durchsucht werden, um festzustellen, welche Daten im Warehouse verfügbar sind.

Sie können außerdem ein Sternschemamodell für die Daten im Warehouse definieren. Ein Sternschema ist ein spezielles Modell, das aus mehreren Dimensionstabellen besteht, die Aspekte eines Unternehmens beschreiben, sowie aus einer Fakttabelle, die die Fakten oder Messungen zum Unternehmen enthält. Für ein Produktionsunternehmen z. B. sind einige Dimensionstabellen die Produkte, die Märkte und die Zeit. Die Fakttabelle enthält Transaktionsinformationen zu den Produkten, die in jedem Gebiet aufgeschlüsselt nach Jahreszeit bestellt wurden.

Zugehörige Konzepte:

- „Warehouse-Schritte" in *Data Warehouse-Zentrale Verwaltung*

Zugehörige Tasks:

- „Schritte und Tasks: Data Warehouse-Zentrale - Hilfe"

Teil 2. Datenbankentwurf

Kapitel 4. Logischer Datenbankentwurf

Das Ziel des Entwurfs einer Datenbank ist es, eine Darstellung der Umgebung zu finden, die leicht verständlich ist und als Basis für zukünftige Erweiterungen dienen kann. Darüber hinaus soll der Aufbau der Datenbank die Wahrung der Konsistenz und der Integrität der Daten unterstützen. Dies wird durch eine Datenbankstruktur erreicht, mit deren Hilfe Redundanz verringert und Anomalien eliminiert werden, die während der Aktualisierung der Datenbank auftreten können.

Der Entwurf einer Datenbank ist kein linearer Prozess. Wahrscheinlich müssen im Laufe des Entwurfs Schritte wiederholt ausgeführt oder überarbeitet werden.

Festlegen der in einer Datenbank aufzuzeichnenden Daten

Der erste Schritt bei der Entwicklung eines Datenbankentwurfs ist die Festlegung, welche Typen von Daten in den Tabellen der Datenbank gespeichert werden sollen. Eine Datenbank umfasst Informationen über die *Entitäten* innerhalb einer Organisation oder Firma sowie über die Beziehungen bzw. Abhängigkeiten zwischen ihnen. In einer relationalen Datenbank werden *Entitäten* als *Tabellen* dargestellt.

Eine *Entität* ist eine Person, ein Objekt oder ein Begriff, über die Informationen gespeichert werden sollen. Einige der Entitäten, die in den Beispieltabellen beschrieben werden, sind zum Beispiel Mitarbeiter, Abteilungen und Projekte.

In der Beispieltabelle EMPLOYEE hat die Entität "EMPLOYEE" (Mitarbeiter) bestimmte *Attribute* oder Merkmale, wie Personalnummer, Name, Abteilung (Work Department) und Gehalt (Salary). Diese Merkmale bilden die *Spalten* EMPNO, FIRSTNME, LASTNAME, WORKDEPT und SALARY der Tabelle.

Ein konkretes *Vorkommen* der Entität "Mitarbeiter" (employee) besteht aus den Werten aller Spalten für einen Mitarbeiter. Jeder Mitarbeiter hat eine eindeutige Personalnummer (EMPNO), die zur Identifizierung dieses Vorkommens der Entität „Mitarbeiter" verwendet werden kann. Jede Zeile in einer Tabelle stellt ein Vorkommen einer Entität oder Relation dar. Die Werte in der ersten Zeile der folgenden Tabelle beschreiben beispielsweise eine Mitarbeiterin namens Haas.

Tabelle 4. Vorkommen der Entität Mitarbeiter und zugehöriger Attribute

EMPNO	FIRSTNME	LASTNAME	WORKDEPT	JOB
000010	Christine	Haas	A00	President
000020	Michael	Thompson	B01	Manager
000120	Sean	O'Connell	A00	Clerk
000130	Dolores	Quintana	C01	Analyst
000030	Sally	Kwan	C01	Manager
000140	Heather	Nicholls	C01	Analyst
000170	Masatoshi	Yoshimura	D11	Designer

Es gibt einen wachsenden Bedarf an Unterstützung für neuere Datenbankanwendungen wie zum Beispiel Multimediaanwendungen. Es kann sinnvoll sein,

Attribute zur Unterstützung von Multimediaobjekten, z. B. Dokumenten, Videodaten oder gemischten Medien, Bildern und Tondaten, zu berücksichtigen.

Innerhalb einer Tabelle hat jede Spalte einer Zeile eine bestimmte Relation oder Beziehung zu allen anderen Spalten dieser Zeile. Einige Relationen, die in den Beispieltabellen zum Ausdruck kommen, sind folgende:

- Mitarbeiter sind Abteilungen zugeordnet.
  - Dolores Quintana gehört zu Abteilung C01 (WORKDEPT).
- Mitarbeiter führen eine Jobbezeichnung.
  - Dolores arbeitet als „Analyst".
- Mitarbeiter leiten Abteilungen.
  - Sally leitet die Abteilung C01.

"Mitarbeiter (Employee) und "Abteilung" (Department) sind Entitäten. Sally Kwan ist Teil eines Vorkommens der Entität "Mitarbeiter", und C01 ist Teil des Vorkommens der Entität "Abteilung". Für dieselben Spalten in jeder Zeile einer Tabelle gelten dieselben Arten von Relationen. Zum Beispiel gibt eine Zeile einer Tabelle die Relation an, dass Sally Kwan die Abteilung C01 leitet. Eine andere Zeile enthält die Relation, dass Sean O'Connell Sachbearbeiter (Clerk) in der Abteilung A00 ist.

Die Informationen innerhalb einer Tabelle sind von der Art der Relation, die dargestellt werden soll, vom Maß der benötigten Flexibilität und von der gewünschten Geschwindigkeit, mit der die Daten abgerufen werden können, abhängig.

Neben der Feststellung, welche Beziehungen zwischen Entitäten innerhalb Ihres Unternehmens in der Datenbank erfasst werden sollen, müssen auch andere Arten von Informationen bestimmt werden, z. B. die Geschäftsregeln, denen die Daten unterliegen.

Zugehörige Konzepte:

- „Beziehungen in einer Datenbank" auf Seite 56
- „Spaltendefinitionen" auf Seite 59

Beziehungen in einer Datenbank

In einer Datenbank können verschiedene Arten von Beziehungen (Relationen) definiert werden. Betrachten Sie zum Beispiel die möglichen Beziehungen zwischen Mitarbeitern (Employees) und Abteilungen (Departments).

Eins-zu-viele- und Viele-zu-eins-Beziehungen

Ein Mitarbeiter kann in nur einer Abteilung arbeiten. Eine solche Beziehung ist *einwertig* für den Mitarbeiter. Auf der anderen Seite kann eine Abteilung mehrere Mitarbeiter haben. Diese Beziehung ist *mehrwertig* für die Abteilungen. Die Relation zwischen Mitarbeiter (einwertig) und Abteilungen (mehrwertig) ist eine *Eins-zu-viele-Beziehung*.

Zur Definition von Tabellen für die einzelnen Eins-zu-viele- und Viele-zu-eins-Beziehungen empfiehlt sich folgende Methode:

1. Fassen Sie alle Beziehungen in Gruppen zusammen, bei denen jeweils die "Viele"-Seite der Relation dieselbe Entität ist.
2. Definieren Sie eine Tabelle für alle Relationen in der Gruppe.

Im folgenden Beispiel ist die Viele-Seite der ersten und zweiten Beziehung "Employees" (Mitarbeiter), so dass eine Tabelle für die Mitarbeiter (Tabelle EMPLOYEE) definiert wird.

Tabelle 5. Viele-zu-eins-Beziehungen

Entität	Beziehung	Entität
Mitarbeiter	sind zugeordnet zu	Abteilungen
Mitarbeiter	arbeiten als	Jobbezeichnung
Abteilungen	berichten an	(Verwaltungs-) Abteilungen

In der dritten Beziehung ist die Entität "Departments" (Abteilungen) auf der Viele-Seite, so dass eine Tabelle DEPARTMENT definiert wird.

Die folgenden Tabellen zeigen diese verschiedenen Beziehungen.

Die Tabelle EMPLOYEE:

EMPNO	WORKDEPT	JOB
000010	A00	President
000020	B01	Manager
000120	A00	Clerk
000130	C01	Analyst
000030	C01	Manager
000140	C01	Analyst
000170	D11	Designer

Die Tabelle DEPARTMENT:

DEPTNO	ADMRDEPT
C01	A00
D01	A00
D11	D01

Viele-zu-viele-Beziehungen

Eine Relation, die in beiden Richtungen mehrwertig ist, wird als eine Viele-zu-viele-Beziehung bezeichnet. Ein Mitarbeiter (Employee) kann an mehr als einem Projekt arbeiten, und ein Projekt kann mehr als einen Mitarbeiter beschäftigen. Auf die Fragen „Woran arbeitet Dolores Quintana?" und „Wer arbeitet an Projekt IF1000?" gibt es jeweils mehrere Antworten. Eine Viele-zu-viele-Beziehung kann in einer Tabelle mit einer Spalte für jede Entität (Mitarbeiter „EMPNO" und Projekt „PROJNO") wie im folgenden Beispiel dargestellt werden.

Die folgende Tabelle zeigt, wie eine Viele-zu-viele-Beziehung (ein Mitarbeiter kann an vielen Projekten arbeiten, und ein Projekt kann viele Mitarbeiter beschäftigen) dargestellt wird:

Die Tabelle EMP_ACT (Mitarbeitertätigkeiten):

EMPNO	PROJNO
000030	IF1000
000030	IF2000
000130	IF1000
000140	IF2000
000250	AD3112

Eins-zu-eins-Beziehung

Eins-zu-eins-Beziehungen sind in beide Richtungen einwertig. Ein Manager leitet eine Abteilung. Eine Abteilung hat nur einen Manager. Auf die Fragen „Wer ist der Manager der Abteilung C01?" und „Welche Abteilung wird von Sally Kwan geleitet?" gibt es jeweils nur eine Antwort. Die Beziehung kann entweder der Tabelle DEPARTMENT oder der Tabelle EMPLOYEE zugeordnet werden. Da alle Abteilungen Manager haben, aber nicht alle Mitarbeiter Manager sind, ist die logische Konsequenz, die Manager der Tabelle DEPARTMENT wie im folgenden Beispiel zuzuordnen.

Die folgende Tabelle zeigt die Darstellung einer Eins-zu-eins-Beziehung.

Die Tabelle DEPARTMENT:

DEPTNO	MGRNO
A00	000010
B01	000020
D11	000060

Sicherstellen, dass gleiche Werte die gleiche Entität darstellen

Sie können die Attribute derselben Menge von Entitäten in mehreren Tabellen beschreiben. Zum Beispiel enthält die Tabelle EMPLOYEE die Nummer der Abteilung (WORKDEPT), der ein Mitarbeiter zugeteilt ist, und die Tabelle DEPARTMENT zeigt, welcher Manager jeder Abteilungsnummer (DEPTNO) zugeordnet ist. Um beide Mengen von Attributen gleichzeitig abzurufen, können die beiden Tabellen über die beiden übereinstimmenden Spalten verknüpft werden, wie im folgenden Beispiel gezeigt wird. Der Wert der Spalte WORKDEPT und der Wert der Spalte DEPTNO stellen dieselbe Entität dar und bilden daher einen *Verknüpfungspfad* zwischen den Tabellen DEPARTMENT und EMPLOYEE.

Die Tabelle DEPARTMENT:

DEPTNO	DEPTNAME	MGRNO	ADMRDEPT
D21	Administration Support	000070	D01

Die Tabelle EMPLOYEE:

EMPNO	FIRSTNAME	LASTNAME	WORKDEPT	JOB
000250	Daniel	Smith	D21	Clerk

Beim Abrufen von Informationen über eine Entität aus mehreren Tabellen muss sichergestellt werden, dass gleiche Werte auch die gleiche Entität bezeichnen. Die verknüpfenden Spalten können unterschiedliche Namen haben (wie WORKDEPT und DEPTNO im vorangegangenen Beispiel), oder sie können auch denselben Namen haben (wie die Spalten mit dem Namen DEPTNO in den Tabellen DEPARTMENT und PROJECT).

Zugehörige Konzepte:

- „Festlegen der in einer Datenbank aufzuzeichnenden Daten" auf Seite 55
- „Spaltendefinitionen" auf Seite 59

Spaltendefinitionen

Eine Spalte in einer relationalen Tabelle wird folgendermaßen definiert:

1. Wählen Sie einen Namen für die Spalte.

   Jede Spalte in einer Tabelle muss einen für diese Tabelle eindeutigen Namen haben.

2. Definieren Sie die Art der Daten, die für die Spalte gültig sein sollen.

   Der *Datentyp* und die *Länge* legen den Typ und die maximale Länge der Daten fest, die als Werte der Spalte gültig sind. Datentypen können aus den im Datenbankmanager vordefinierten Typen ausgewählt oder vom Benutzer als so genannte benutzerdefinierte Typen selbst erstellt werden.

   Kategorien von Datentypen sind zum Beispiel: numerische Daten, Zeichenfolge, Doppelbytezeichenfolge (oder Grafikzeichenfolge), Datum/Uhrzeit und Binärzeichenfolge.

   Der Datentyp *großes Objekt* (LOB) unterstützt Multimedia-Objekte wie Dokumente, Video-, Abbild- und Tondaten. Diese Objekte werden mit Hilfe der folgenden Datentypen implementiert:

   - Eine Zeichenfolge des Typs *großes Binärobjekt* (BLOB). Beispiele für BLOBs sind Fotografien von Mitarbeitern, Stimmenaufnahmen und Videoaufnahmen.
   - Eine Zeichenfolge des Typs *großes Zeichenobjekt* (CLOB), in der eine Folge von Zeichen entweder aus Einzelbyte- oder Mehrbytezeichen oder aus einer Kombination aus beidem bestehen kann. Ein Beispiel für ein CLOB ist ein Lebenslauf eines Mitarbeiters.
   - Eine Zeichenfolge des Typs *Doppelbytezeichen für großes Objekt* (DBCLOB), in der eine Folge von Zeichen aus Doppelbytezeichen besteht. Ein Beispiel für ein DBCLOB ist ein Lebenslauf in japanischer Sprache.

   Ein *benutzerdefinierter Datentyp* (UDT - User-Defined Type) ist ein Datentyp, der von einem vorhandenen Typ abgeleitet ist. Es können Datentypen definiert werden, die von vorhandenen Typen abgeleitet sind und über ähnliche Merkmale wie die vorhandenen Typen verfügen, die jedoch als separat und nicht kompatibel behandelt werden sollen.

   Ein *strukturierter Typ* ist ein benutzerdefinierter Typ, dessen Struktur in der Datenbank definiert ist. Er enthält eine Reihe benannter *Attribute*, die jeweils über einen Datentyp verfügen. Ein strukturierter Typ kann als *untergeordneter Typ* eines anderen strukturierten Typs definiert sein, der dementsprechend als zugehöriger *übergeordneter Typ* bezeichnet wird. Ein untergeordneter Typ übernimmt alle Attribute des dazugehörigen übergeordneten Typs und kann zusätzlich über weitere Attribute verfügen. Die Gruppe strukturierter Typen, die zu einem gemeinsamen übergeordneten Typ gehören, wird als *Typhierarchie*

bezeichnet. Ein übergeordneter Typ, der keinem anderen Typ untergeordnet ist, wird als *Stammtyp* der Typhierarchie bezeichnet.

Ein strukturierter Typ kann als Typ einer Tabelle oder einer Sicht verwendet werden. Die Namen und Datentypen der Attribute der strukturierten Typen werden zusammen mit der Objektkennung als Namen und Datentypen für die Spalten dieser *typisierten Tabelle* oder *typisierten Sicht* verwendet. Zeilen einer typisierten Tabelle oder typisierten Sicht können als Darstellung von Exemplaren des strukturierten Typs verstanden werden.

Ein strukturierter Typ kann nicht als Datentyp für eine Spalte einer Tabelle oder Sicht verwendet werden. Außerdem besteht keine Möglichkeit, ein vollständiges Exemplar eines strukturierten Typs in eine Hostvariable eines Anwendungsprogramms zu übernehmen.

Ein *Referenztyp* ist ein Begleittyp zum strukturierten Typ. Wie ein einzigartiger Datentyp ist auch der Referenztyp ein Skalartyp, dessen Darstellung mit der Darstellung eines integrierten Datentyps identisch ist. Diese Darstellung wird von allen Typen der Typhierarchie gemeinsam benutzt. Die Darstellung des Referenztyps wird beim Erstellen des Stammtyps einer Typhierarchie definiert. Bei Verwendung eines Referenztyps wird ein strukturierter Typ als Parameter des Typs angegeben. Dieser Parameter wird als *Zieltyp* der Referenz bezeichnet.

Das Ziel einer Referenz ist immer eine Zeile in einer typisierten Tabelle oder Sicht. Bei Verwendung eines Referenztyps kann ein *Bereich* (Scope) definiert werden. Der Bereich identifiziert eine Tabelle (die *Zieltabelle*) oder eine Sicht (die *Zielsicht*), die die Zielzeile eines Referenzwerts enthält. Die Zieltabelle oder -sicht muss denselben Typ aufweisen wie der Zieltyp des Referenztyps. Ein Exemplar eines Referenztyps mit Bereich identifiziert eindeutig eine Zeile in einer typisierten Tabelle oder Sicht, die als *Zielzeile* dieses Referenztyps bezeichnet wird.

Eine *benutzerdefinierte Funktion* (UDF - User-Defined Function) kann zu zahlreichen Zwecken verwendet werden, darunter auch zum Aufrufen von Routinen, die Vergleiche oder Umwandlungen zwischen benutzerdefinierten Datentypen ermöglichen. Benutzerdefinierte Funktionen erweitern und ergänzen die Unterstützung, die von integrierten SQL-Funktionen zur Verfügung gestellt wird, und können überall dort verwendet werden, wo auch integrierte Funktionen verwendet werden können. Es gibt zwei Arten benutzerdefinierter Funktionen:

- Externe Funktionen, die in einer Programmiersprache geschrieben sind
- Quellenfunktionen, die zum Aufrufen anderer benutzerdefinierter Funktionen verwendet werden

Beispielsweise gibt es die beiden numerischen Datentypen der europäischen Schuhgröße und der amerikanischen Schuhgröße. Beide Typen stellen Schuhgrößen dar, aber sie sind nicht miteinander kompatibel, da die Maßeinheiten verschieden sind und nicht direkt miteinander verglichen werden können. Eine benutzerdefinierte Funktion kann aufgerufen werden, um die Maßangaben von einer Schuhgröße in eine andere umzuwandeln.

3. Geben Sie an, welche Spalten eventuell Standardwerte enthalten müssen.

   Einige Spalten können nicht in allen Zeilen sinnvolle Werte enthalten. Dafür kann es folgende Gründe geben:

   - Ein Spaltenwert ist für die Zeile nicht zutreffend.

     Zum Beispiel kann eine Spalte, die die Mittelinitialen von Mitarbeiternamen enthält, keinen sinnvollen Wert enthalten, wenn der in der Zeile eingetragene Mitarbeiter keine Mittelinitiale hat.
   - Ein Wert ist zwar zutreffend, aber er ist noch nicht bekannt.

Ein Beispiel wäre die Spalte MGRNO (Managernummer), die vorübergehend keine gültige Managernummer enthält, weil der vorige Manager versetzt wurde, aber bislang noch kein neuer Manager bestimmt wurde.

In beiden Fällen haben Sie die Wahl, einen Nullwert (einen speziellen Wert, der angibt, dass der Spaltenwert unbekannt oder unzutreffend ist) zuzulassen oder einen vom Nullwert abweichenden Standardwert durch den Datenbankmanager oder die Anwendung zuordnen zu lassen.

Primärschlüssel

Ein *Schlüssel* ist eine Gruppe von Spalten, mit deren Hilfe eine Zeile oder Zeilen identifiziert werden oder auf sie zugegriffen wird. Der Schlüssel wird in der Beschreibung einer Tabelle, eines Index oder einer referenziellen Integritätsbedingung angegeben. Dieselbe Spalte kann Teil von mehr als einem Schlüssel sein.

Ein *eindeutiger Schlüssel* ist ein Schlüssel mit der Integritätsbedingung, dass nicht zwei seiner Werte gleich sein dürfen. Die Spalten eines eindeutigen Schlüssels können keine Nullwerte enthalten. Zum Beispiel kann eine Spalte mit der Personalnummer als eindeutiger Schlüssel definiert werden, da jeder Wert in der Spalte nur einen Mitarbeiter identifiziert. Innerhalb eines Unternehmens können nicht zwei Mitarbeiter dieselbe Personalnummer haben.

Der Mechanismus, mit dem die Eindeutigkeit sichergestellt wird, wird als *eindeutiger Index* bezeichnet. Der eindeutige Index einer Tabelle ist eine Spalte oder eine geordnete Gruppe von Spalten, deren Werte jeweils eine Zeile eindeutig identifizieren (funktionell bestimmen). Ein eindeutiger Index kann Nullwerte enthalten.

Der *Primärschlüssel* ist einer der für eine Tabelle definierten eindeutigen Schlüssel, der jedoch als Schlüssel mit der höchsten Priorität ausgewählt ist. Es kann nur einen Primärschlüssel für eine Tabelle geben.

Für den Primärschlüssel wird automatisch ein *Primärindex* erstellt. Der Primärindex wird vom Datenbankmanager zum effizienten Zugriff auf Tabellenzeilen verwendet und ermöglicht es dem Datenbankmanager, die Eindeutigkeit des Primärschlüssels zu erhalten. (Sie können außerdem Indizes für Spalten definieren, die nicht zum Primärschlüssel gehören, um bei der Verarbeitung von Abfragen einen effizienten Zugriff auf Daten bieten zu können.)

Wenn die Tabelle keinen „natürlichen" eindeutigen Schlüssel besitzt, oder wenn zur Unterscheidung eindeutiger Zeilen die Eingangsfolge verwendet wird, kann auch die Verwendung einer Zeitmarke als Bestandteil des Schlüssels sinnvoll sein.

Primärschlüssel für einige der Beispieltabellen sind folgende:

Tabelle	**Schlüsselspalte**
Tabelle EMPLOYEE	EMPNO
Tabelle DEPARTMENT	DEPTNO
Tabelle PROJECT	PROJNO

Das folgende Beispiel zeigt einen Teil der Tabelle PROJECT mit ihrem Primärschlüssel.

Tabelle 6. Ein Primärschlüssel in der Tabelle PROJECT

PROJNO (Primärschlüssel)	PROJNAME	DEPTNO
MA2100	Weld Line Automation	D01
MA2110	Weld Line Programming	D11

Wenn jede Spalte einer Tabelle doppelte Werte enthält, kann kein Primärschlüssel aus nur einer Spalte definiert werden. Ein Schlüssel, der aus mehreren Spalten besteht, wird als *zusammengesetzter Schlüssel* bezeichnet. Die Kombination der Spaltenwerte muss eine Entität eindeutig definieren. Wenn ein zusammengesetzter Schlüssel nicht einfach definiert werden kann, ist eventuell die Erstellung einer neuen Spalte mit eindeutigen Werten in Betracht zu ziehen.

Das folgende Beispiel zeigt einen Primärschlüssel, der mehrere Spalten (einen zusammengesetzten Schlüssel) enthält:

Tabelle 7. Ein zusammengesetzter Primärschlüssel in der Tabelle EMP_ACT

EMPNO (Primär-schlüssel)	PROJNO (Primär-schlüssel)	ACTNO (Primär-schlüssel)	EMPTIME	EMSTDATE (Primär-schlüssel)
000250	AD3112	60	1.0	1982-01-01
000250	AD3112	60	.5	1982-02-01
000250	AD3112	70	.5	1982-02-01

Bestimmen von Schlüsselkandidatenspalten

Als Schlüsselkandidat bietet sich jeweils die kleinste Anzahl von Spalten an, die eine Entität eindeutig definieren. Es kann mehrere Schlüsselkandidaten geben. In Tabelle 8 gibt es scheinbar mehrere Schlüsselkandidaten. Die Spalten EMPNO, PHONENO und LASTNAME identifizieren den Mitarbeiter eindeutig.

Tabelle 8. Tabelle EMPLOYEE

EMPNO (Primär-schlüssel)	FIRSTNAME	LASTNAME	WORKDEPT (Fremd-schlüssel)	PHONENO
000010	Christine	Haas	A00	3978
000030	Sally	Kwan	C01	4738
000060	Irving	Stern	D11	6423
000120	Sean	O'Connell	A00	2167
000140	Heather	Nicholls	C01	1793
000170	Masatoshi	Yoshimura	D11	2890

Bei der Auswahl eines Primärschlüssels aus einer Gruppe möglicher Schlüssel sollten die Kriterien Permanenz, Eindeutigkeit und Stabilität beachtet werden.

- Permanenz heißt, dass der Primärschlüssel für die Zeile stets vorhanden ist.
- Eindeutigkeit heißt, dass der Schlüsselwert für eine Zeile sich von den Schlüsselwerten aller anderen Zeilen unterscheidet.
- Stabilität heißt, dass die Werte für Primärschlüssel nie geändert werden.

Von den drei möglichen Schlüsselspalten im genannten Beispiel erfüllt nur die Spalte EMPNO alle Kriterien. Es kann zum Beispiel vorkommen, dass ein Mitarbeiter keine Telefonnummer (PHONENO) hat, wenn er in die Firma eintritt. Nachnamen (LASTNAME) können sich ändern und sind, wenn auch vielleicht zu einem bestimmten Zeitpunkt, so doch nicht unbedingt immer eindeutig. Die Personalnummer (EMPNO) ist die beste Wahl für den Primärschlüssel. Ein Mitarbeiter erhält nur einmal eine eindeutige Nummer, und diese Nummer wird in der Regel nicht geändert, solange der Mitarbeiter in der Firma bleibt. Da jeder Mitarbeiter unbedingt eine Nummer haben muss, gilt auch das Kriterium der Permanenz für die Werte dieser Spalte.

Zugehörige Konzepte:

• „Identitätsspalten" auf Seite 63

Identitätsspalten

Eine *Identitätsspalte* stellt für DB2® Universal Database (DB2 UDB) eine Methode dar, automatisch einen eindeutigen numerischen Wert für jede Zeile in einer Tabelle zu generieren. Eine Tabelle kann eine einzelne Spalte besitzen, die mit dem Attribut IDENTITY definiert ist. Beispiele für IDENTITY-Spalten sind Auftragsnummern, Personalnummern, Bestandsnummern und Vorfallsnummern.

Werte für eine IDENTITY-Spalte können immer oder standardmäßig generiert werden.

- Die Eindeutigkeit einer IDENTITY-Spalte, die als *immer generiert* (generated always) definiert ist, wird von DB2 UDB garantiert. Die Werte der Spalte werden immer von DB2 UDB generiert. Die Angabe eines expliziten Werts durch Anwendungen wird nicht zugelassen.
- Eine IDENTITY-Spalte, die als *standardmäßig generiert* (generated by default) definiert ist, gibt Anwendungen eine Möglichkeit, einen Wert in die IDENTITY-Spalte explizit einzugeben. Wenn kein Wert angegeben wird, generiert DB2 UDB einen, kann jedoch die Eindeutigkeit des Werts in diesem Fall nicht garantieren. DB2 UDB garantiert die Eindeutigkeit nur für die Gruppe von Werten, die von DB2 generiert wird. Die Definition *standardmäßig generiert* ist zur Verwendung bei Datenweitergaben (Data Propagation), bei der der Inhalt einer vorhandenen Tabelle kopiert wird, oder zum Laden von Inhalt aus einer Tabelle bzw. erneutes Laden in eine Tabelle gedacht.

IDENTITY-Spalten eignen sich ideal zur Generierung eindeutiger Primärschlüsselwerte. Anwendungen können IDENTITY-Spalten nutzen, um Problemen beim gemeinsamen Zugriff und Leistungsproblemen vorzubeugen, die entstehen können, wenn eine Anwendung einen eigenen eindeutigen Zähler außerhalb der Datenbank generiert. Zum Beispiel besteht eine gängige Implementierung auf Anwendungsebene darin, eine 1-zeilige Tabelle mit einem Zähler zu pflegen. Jede Transaktion sperrt diese Tabelle, erhöht den Wert des Zählers und schreibt die Änderung fest. Das heißt, zu einem gegebenen Zeitpunkt kann nur eine Transaktion den Wert des Zählers erhöhen. Wenn im Unterschied dazu der Zähler durch eine IDENTITY-Spalte realisiert wird, können wesentliche höhere Ebenen des gemeinsamen Zugriffs erreicht werden, weil der Zähler nicht durch Transaktionen gesperrt wird. Eine nicht festgeschriebene Transaktion, die den Wert des Zählers erhöht hat, hindert nachfolgende Transaktionen nicht daran, den Zähler ebenfalls zu erhöhen.

Der Wert des Zählers für die IDENTITY-Spalte wird unabhängig von der Transaktion erhöht (oder verringert). Wenn eine bestimmte Transaktion einen IDENTITY-

Zähler zweimal erhöht, stellt diese Transaktion möglicherweise einen Abstand zwischen den beiden generierten Nummern fest, weil vielleicht andere Transaktionen zur gleichen Zeit denselben IDENTITY-Zähler ebenfalls erhöhen (d. h. Zeilen in die gleiche Tabelle einfügen). Wenn eine Anwendung unbedingt einen aufeinander folgenden Bereich von Nummern erhalten muss, sollte die Anwendung eine exklusive Sperre für die Tabelle definieren, in der die IDENTITY-Spalte enthalten ist. Diese Entscheidung muss jedoch gegen die daraus folgende Einbuße an gemeinsamen Zugriff abgewogen werden. Weiterhin ist es möglich, dass eine bestimmte IDENTITY-Spalte so aussehen kann, als habe sie Sprünge in den Nummern generiert, weil eine Transaktion, die einen Wert für die IDENTITY-Spalte generierte, rückgängig gemacht wurde, oder weil die Datenbank, die einen Bereich von Werten in einem Cache gespeichert hatte, deaktiviert wurde, bevor alle im Cache gespeicherten Werte zugeordnet werden konnten.

Die fortlaufenden Nummern, die durch die IDENTITY-Spalte generiert werden, besitzen die folgenden zusätzlichen Merkmale:

- Die Werte können jeden exakten numerischen Datentyp ohne Kommastellen, d. h. SMALLINT, INTEGER, BIGINT oder DECIMAL mit null Kommastellen, besitzen. (Gleitkommazahlen mit einfacher und doppelter Genauigkeit werden als angenäherte numerische Datentypen bezeichnet.)
- Aufeinander folgende Werte können sich um ein beliebiges angegebenes ganzzahliges Inkrement unterscheiden. Das Standardinkrement ist 1.
- Der Wert des Zählers für die IDENTITY-Spalte ist wiederherstellbar. Wenn ein Fehler auftritt, wird der Wert des Zählers aus den Protokollen rekonstruiert, wodurch sichergestellt wird, dass weiterhin eindeutige Werte generiert werden.
- Werte von IDENTITY-Spalten können im Cache gespeichert werden, um eine bessere Leistung zu erzielen.

Zugehörige Konzepte:

- „Primärschlüssel" auf Seite 61

Normalisierung

Eine *Normalisierung* hilft bei der Vermeidung von Redundanzen und Inkonsistenzen in Tabellendaten. Sie bezeichnet den Prozess der Reduzierung von Tabellen auf eine Menge von Spalten, in der alle Spalten, die selbst zu keinem Schlüssel gehören, von der Primärschlüsselspalte abhängig sind. Wenn dies nicht der Fall ist, können die Daten durch Aktualisierungen inkonsistent werden.

In diesem Abschnitt werden kurz die Kriterien für die erste, zweite, dritte und vierte Normalform vorgestellt. Die fünfte Normalform einer Tabelle, die in zahlreichen Veröffentlichungen zum Entwurf von Datenbanken dargestellt wird, ist hier nicht beschrieben.

Form	Beschreibung
Erste	An jeder Zeilen- und Spaltenposition in der Tabelle existiert nur *ein* Wert und zu keiner Zeit eine Gruppe von Werten.
Zweite	Jede Spalte, die nicht Teil des Schlüssels ist, ist vom Schlüssel abhängig.
Dritte	Jede Spalte ohne Schlüsselfunktion ist von anderen Spalten ohne Schlüsselfunktion unabhängig und nur vom Schlüssel abhängig.
Vierte	Keine Zeile enthält zwei oder mehr unabhängige, mehrwertige Fakten über eine Entität.

Erste Normalform

Eine Tabelle ist in der *ersten Normalform*, wenn sich in jeder Zelle nur ein Wert und nie eine Gruppe von Werten befindet. Eine Tabelle in der ersten Normalform erfüllt nicht unbedingt auch die Kriterien für höhere Normalformen.

Die folgende Tabelle verstößt beispielsweise gegen die Kriterien der ersten Normalform, weil in der Spalte LAGER mehrere Werte für jedes Vorkommen von ARTIKEL auftreten.

Tabelle 9. Tabelle, die die erste Normalform verletzt

ARTIKEL (Primärschlüssel)	LAGER
P0010	Lager A, Lager B, Lager C
P0020	Lager B, Lager D

Das folgende Beispiel zeigt die gleiche Tabelle in der ersten Normalform.

Tabelle 10. Tabelle in der ersten Normalform

ARTIKEL (Primärschlüssel)	LAGER (Primärschlüssel)	BESTAND
P0010	Lager A	400
P0010	Lager B	543
P0010	Lager C	329
P0020	Lager B	200
P0020	Lager D	278

Zweite Normalform

Eine Tabelle ist in der *zweiten Normalform*, wenn jede Spalte, die nicht zum Schlüssel gehört, vom *gesamten* Schlüssel abhängig ist.

Die Kriterien der zweiten Normalform sind nicht erfüllt, wenn eine Spalte ohne Schlüsselfunktion von einem *Teil* eines zusammengesetzten Schlüssels abhängig ist, wie im folgenden Beispiel zu sehen ist:

Tabelle 11. Tabelle, die die zweite Normalform verletzt

ARTIKEL (Primärschlüssel)	LAGER (Primärschlüssel)	BESTAND	LAGER_ADRESSE
P0010	Lager A	400	Amselfelderstraße 24
P0010	Lager B	543	Industrieweg 6
P0010	Lager C	329	Kastanienallee 71
P0020	Lager B	200	Industrieweg 6
P0020	Lager D	278	Ulmenstraße 8

Der Primärschlüssel ist ein zusammengesetzter Schlüssel, der aus den Spalten ARTIKEL und LAGER besteht. Da die Spalte LAGER_ADRESSE jedoch nur vom Wert der Spalte LAGER abhängig ist, verstößt die Tabelle gegen die Regel der zweiten Normalform.

Dieser Tabellenentwurf hat folgende Nachteile:

- Die Lageradresse wird in jeden Datensatz für einen Artikel, der in dem entsprechenden Lagerhaus gelagert wird, wiederholt.
- Wenn die Adresse eines Lagerhauses geändert wird, muss jede Zeile, die sich auf einen Artikel in diesem Lagerhaus bezieht, aktualisiert werden.
- Aufgrund dieser Redundanz könnten die Daten inkonsistent werden, wenn für dasselbe Lagerhaus in verschiedenen Datensätzen unterschiedliche Adressen auftreten.
- Wenn zu einem Zeitpunkt keine Artikel in dem Lagerhaus gelagert werden, ist vielleicht keine Zeile mehr vorhanden, in der die Adresse des Lagerhauses gespeichert werden kann.

Die Lösung besteht in einer Aufspaltung der Tabelle in die beiden folgenden Tabellen:

Tabelle 12. Die Tabelle LAGERBESTAND erfüllt die zweite Normalform

ARTIKEL (Primärschlüssel)	LAGER (Primärschlüssel)	BESTAND
P0010	Lager A	400
P0010	Lager B	543
P0010	Lager C	329
P0020	Lager B	200
P0020	Lager D	278

Tabelle 13. Die Tabelle LAGERHAUS erfüllt die zweite Normalform

LAGER (Primärschlüssel)	LAGER_ADRESSE
Lager A	Amselfelderstraße 24
Lager B	Industrieweg 6
Lager C	Kastanienallee 71
Lager D	Ulmenstraße 8

Es muss beachtet werden, dass die Erstellung zweier Tabellen in der zweiten Normalform Auswirkungen auf die Leistung haben kann. Anwendungen, die Berichte über den Lagerort von Artikeln erstellen, müssen beide Tabellen verknüpfen, um die relevanten Informationen abrufen zu können.

Dritte Normalform

Eine Tabelle ist in der dritten Normalform, wenn jede Spalte ohne Schlüsselfunktion von anderen Spalten ohne Schlüsselfunktion unabhängig und nur vom Schlüssel abhängig ist.

Die erste Tabelle des folgenden Beispiels enthält die Spalten EMPNO und WORKDEPT. Nehmen Sie an, dass die Spalte DEPTNAME hinzugefügt wird (siehe Tabelle 15 auf Seite 67). Die neue Spalte ist von der Spalte WORKDEPT abhängig, während der Primärschlüssel nur die Spalte EMPNO umfasst. Die Tabelle verstößt nun gegen die Kriterien der dritten Normalform. Durch Ändern der Spalte DEPTNAME für einen einzelnen Mitarbeiter, z. B. John Parker, wird nicht auch der Abteilungsname für andere Mitarbeiter dieser Abteilung geändert. Beachten Sie, dass es nun zwei unterschiedliche Abteilungsnamen (DEPTNAME) für Abteilungsnummer (WORKDEPT) E11 gibt. Die sich daraus ergebende Inkonsistenz zeigt sich in der aktualisierten Version der Tabelle.

Tabelle 14. Nicht normalisierte Tabelle EMPLOYEE_DEPARTMENT vor der Aktualisierung

EMPNO (Primärschlüssel)	FIRSTNAME	LASTNAME	WORKDEPT	DEPTNAME
000290	John	Parker	E11	Operations
000320	Ramlal	Mehta	E21	Software Support
000310	Maude	Setright	E11	Operations

Tabelle 15. Nicht normalisierte Tabelle EMPLOYEE_DEPARTMENT nach der Aktualisierung. Die Daten der Tabelle sind nicht mehr konsistent.

EMPNO (Primärschlüssel)	FIRSTNAME	LASTNAME	WORKDEPT	DEPTNAME
000290	John	Parker	E11	Installation Mgmt
000320	Ramlal	Mehta	E21	Software Support
000310	Maude	Setright	E11	Operations

Die Tabelle kann normalisiert werden, indem eine neue Tabelle mit Spalten für WORKDEPT und DEPTNAME erstellt wird. Eine Aktualisierung wie das Ändern des Abteilungsnamens (DEPTNAME) ist nun wesentlich einfacher. Es muss nur eine Tabelle aktualisiert werden.

Eine SQL-Abfrage, mit der der Abteilungsname zusammen mit dem Namen des Mitarbeiters abgerufen wird, ist dagegen etwas komplizierter, da die beiden Tabellen verknüpft werden müssen. Sie wird wahrscheinlich eine etwas längere Zeit zur Ausführung benötigen als eine Abfrage auf eine einzelne Tabelle. Ferner ist zusätzlicher Speicherplatz erforderlich, da die Spalte WORKDEPT (Abteilung) in beiden Tabellen enthalten sein muss.

Die folgenden Tabellen werden als Ergebnis dieser Normalisierung definiert:

Tabelle 16. Tabelle EMPLOYEE nach Normalisierung der Tabelle EMPLOYEE_DEPARTMENT

EMPNO (Primärschlüssel)	FIRSTNAME	LASTNAME	WORKDEPT
000290	John	Parker	E11
000320	Ramlal	Mehta	E21
000310	Maude	Setright	E11

Tabelle 17. Tabelle DEPARTMENT nach Normalisierung der Tabelle EMPLOYEE_DEPARTMENT

DEPTNO (Primärschlüssel)	DEPTNAME
E11	Operations
E21	Software Support

Vierte Normalform

Eine Tabelle ist in der vierten Normalform, wenn keine Zeile zwei oder mehr unabhängige, mehrwertige Fakten bezüglich einer Entität enthält.

Betrachten Sie zum Beispiel folgende Entitäten: Mitarbeiter (Employee), Fachkenntnisse (Skill) und Sprachen (Language). Ein Mitarbeiter kann über Fachkenntnisse auf mehreren Gebieten verfügen und mehrere Sprachen beherrschen. Es gibt zwei Beziehungen, eine zwischen Mitarbeitern und Fachkenntnissen und eine zwischen Mitarbeitern und Sprachen. Eine Tabelle verstößt gegen die Kriterien der vierten Normalform, wenn sie beide Beziehungen darstellt, wie im folgenden Beispiel gezeigt:

Tabelle 18. Tabelle, die die vierte Normalform verletzt

EMPNO (Primärschlüssel)	SKILL (Primärschlüssel)	LANGUAGE (Primärschlüssel)
000130	Data Modelling	English
000130	Database Design	English
000130	Application Design	English
000130	Data Modelling	Spanish
000130	Database Design	Spanish
000130	Application Design	Spanish

Statt dessen sollten die beiden Beziehungen in zwei Tabellen dargestellt werden:

Tabelle 19. Tabelle EMPLOYEE_SKILL in der vierten Normalform

EMPNO (Primärschlüssel)	SKILL (Primärschlüssel)
000130	Data Modelling
000130	Database Design
000130	Application Design

Tabelle 20. Tabelle EMPLOYEE_LANGUAGE in der vierten Normalform

EMPNO (Primärschlüssel)	LANGUAGE (Primärschlüssel)
000130	English
000130	Spanish

Wenn jedoch die Attribute gegenseitige Abhängigkeiten zeigen, (d. h., der Mitarbeiter verwendet bestimmte Sprachen nur im Zusammenhang mit bestimmten Fachgebieten), dann sollte die Tabelle *nicht* geteilt werden.

Eine gute Strategie bei der Entwicklung eines Datenbankentwurfs besteht darin, alle Daten in Tabellen in der vierten Normalform anzuordnen und anschließend zu entscheiden, ob das Ergebnis eine akzeptable Leistung liefert. Ist dies nicht der Fall, können Sie die Daten in Tabellen in der dritten Normalform neu anordnen und die Leistung erneut begutachten.

Integritätsbedingungen

Eine *Integritätsbedingung* ist eine Regel, die vom Datenbankmanager angewendet wird.

Es gibt vier Typen von Integritätsbedingungen:

- Eine *eindeutige Integritätsbedingung* ist eine Regel, die untersagt, dass in einer oder mehreren Spalten innerhalb einer Tabelle doppelte Werte auftreten. Eindeutige Integritätsbedingungen werden in Form von eindeutigen Schlüsseln und Primärschlüsseln unterstützt. Zum Beispiel kann eine eindeutige Integritätsbedingung für die Lieferantenkennung einer Lieferantentabelle definiert werden, um sicherzustellen, dass nicht ein und dieselbe Kennung zwei Lieferanten zugewiesen wird.
- Eine *referenzielle Integritätsbedingung* ist eine logische Regel über Werte in einer oder mehreren Spalten in mindestens einer Tabelle. Zum Beispiel enthält eine Gruppe von Tabellen gemeinsame Informationen über die Lieferanten einer Firma. Gelegentlich ändert sich der Name eines Lieferanten. Sie können eine referenzielle Integritätsbedingung definieren, die besagt, dass die Kennung (ID) des Lieferanten in einer Tabelle mit einer Lieferanten-ID in den Lieferanteninformationen übereinstimmen muss. Diese Integritätsbedingung verhindert Einfüge-, Aktualisierungs- oder Löschoperationen, die ansonsten zu fehlenden Lieferanteninformationen führen würden.
- Eine *Prüfung auf Integritätsbedingung in Tabellen* definiert Einschränkungen für Daten, die einer bestimmten Tabelle hinzugefügt werden. Zum Beispiel kann eine Prüfung auf Integritätsbedingung in einer Tabelle gewährleisten, dass das Gehaltsniveau eines Mitarbeiters stets mindestens $20.000 beträgt, wenn Gehaltsdaten in einer Tabelle mit Personalinformationen hinzugefügt oder aktualisiert werden.
- Eine *informative Integritätsbedingung* ist eine Regel, die vom SQL-Compiler verwendet werden kann, jedoch vom Datenbankmanager nicht umgesetzt wird.

Referenzielle Integritätsbedingungen und Prüfungen auf Integritätsbedingungen in Tabellen können aktiviert oder inaktiviert werden. Allgemein empfiehlt sich, die Umsetzung einer Integritätsbedingung zum Beispiel dann auszuschalten, wenn große Datenmengen in eine Datenbank geladen werden.

Eindeutige Integritätsbedingungen

Eine *eindeutige Integritätsbedingung* ist die Regel, dass die Werte eines Schlüssels nur dann gültig sind, wenn sie innerhalb einer Tabelle eindeutig sind. Eindeutige Integritätsbedingungen sind optional und können in der Anweisung CREATE TABLE oder ALTER TABLE durch die Klausel PRIMARY KEY oder UNIQUE definiert werden. Die Spalten, die in einer eindeutigen Integritätsbedingung angegeben werden, müssen mit NOT NULL (keine Nullwerte möglich) definiert werden. Der Datenbankmanager stellt die Eindeutigkeit des Schlüssels bei Änderungen an den Spalten der eindeutigen Integritätsbedingung mit Hilfe eines eindeutigen Index sicher.

Eine Tabelle kann eine beliebige Anzahl von Integritätsbedingungen für Eindeutigkeit haben, wobei maximal eine eindeutige Integritätsbedingung als Primärschlüssel definiert sein kann. Eine Tabelle kann nicht mehr als eine eindeutige Integritätsbedingung für dieselbe Gruppe von Spalten haben.

Eine eindeutige Integritätsbedingung, auf die im Fremdschlüssel einer referenziellen Integritätsbedingung verwiesen wird, heißt *übergeordneter Schlüssel*.

Wenn eine eindeutige Integritätsbedingung in einer Anweisung CREATE TABLE definiert wird, wird vom Datenbankmanager automatisch ein eindeutiger Index erstellt und als systemerforderlicher Primärindex bzw. eindeutiger Index gekennzeichnet.

Wenn eine eindeutige Integritätsbedingung in einer Anweisung ALTER TABLE definiert wird und ein Index für die gleichen Spalten vorhanden ist, wird dieser vorhandene Index als eindeutig und systemerforderlich gekennzeichnet. Falls kein solcher Index vorhanden ist, wird vom Datenbankmanager automatisch ein eindeutiger Index erstellt und als systemerforderlicher Primärindex bzw. eindeutiger Index gekennzeichnet.

Beachten Sie, dass zwischen dem Definieren einer eindeutigen Integritätsbedingung und dem Erstellen eines eindeutigen Index ein Unterschied besteht. Zwar dienen beide zur Erhaltung der Eindeutigkeit, jedoch lässt ein eindeutiger Index Spalten mit möglichen Nullwerten zu und kann in der Regel nicht als übergeordneter Schlüssel verwendet werden.

Referenzielle Integritätsbedingungen

Als *referenzielle Integrität* wird der Status einer Datenbank bezeichnet, in der alle Werte aller Fremdschlüssel gültig sind. Ein *Fremdschlüssel* ist eine Spalte oder eine Gruppe von Spalten in einer Tabelle, deren Werte mit mindestens einem Wert des Primärschlüssels oder eines eindeutigen Schlüssels einer Zeile in der übergeordneten Tabelle übereinstimmen müssen. Eine *referenzielle Integritätsbedingung* ist die Regel, dass die Werte des Fremdschlüssels nur dann gültig sind, wenn eine der folgenden Bedingungen gilt:

- Sie treten als Werte eines übergeordneten Schlüssels auf.
- Eine Komponente des Fremdschlüssels ist NULL.

Die Tabelle, die den übergeordneten Schlüssel enthält, wird als *übergeordnete Tabelle* der referenziellen Integritätsbedingung bezeichnet, während die Tabelle, die den Fremdschlüssel enthält, als *abhängige Tabelle* dieser Tabelle bezeichnet wird.

Referenzielle Integritätsbedingungen sind optional und können in der Anweisung CREATE TABLE oder ALTER TABLE definiert werden. Referenzielle Integritätsbedingungen werden durch den Datenbankmanager bei der Ausführung der Anweisungen INSERT, UPDATE, DELETE, ALTER TABLE, ADD CONSTRAINT und SET INTEGRITY angewendet.

Referenzielle Integritätsbedingungen mit der Lösch- oder Aktualisierungsregel RESTRICT werden vor allen anderen referenziellen Integritätsbedingungen angewendet. Referenzielle Integritätsbedingungen mit der Lösch- oder Aktualisierungsregel NO ACTION verhalten sich in den meisten Fällen wie unter der Regel RESTRICT.

Beachten Sie, dass referenzielle Integritätsbedingungen, Prüfungen auf Integritätsbedingungen in Tabellen und Auslöser kombiniert werden können.

Die Regeln der referenziellen Integrität lassen sich durch die folgenden Konzepte und die folgende Terminologie definieren:

Übergeordneter Schlüssel
: Ein Primärschlüssel bzw. ein eindeutiger Schlüssel einer referenziellen Integritätsbedingung.

Übergeordnete Zeile
Eine Zeile, die mindestens eine von ihr abhängige Zeile hat.

Übergeordnete Tabelle
Eine Tabelle, die den übergeordneten Schlüssel einer referenziellen Integritätsbedingung enthält. Eine Tabelle kann für beliebig viele referenzielle Integritätsbedingungen die übergeordnete Tabelle sein. Eine Tabelle, die in einer referenziellen Integritätsbedingung übergeordnete Tabelle ist, kann gleichzeitig abhängige Tabelle in einer referenziellen Integritätsbedingung sein.

Abhängige Tabelle
Eine Tabelle, die in ihrer Definition mindestens eine referenzielle Integritätsbedingung enthält. Eine Tabelle kann für beliebig viele referenzielle Integritätsbedingungen eine abhängige Tabelle sein. Eine Tabelle, die in einer referenziellen Integritätsbedingung eine abhängige Tabelle ist, kann gleichzeitig die übergeordnete Tabelle in einer referenziellen Integritätsbedingung sein.

Untergeordnete Tabelle
Eine Tabelle ist eine untergeordnete Tabelle von Tabelle T, wenn sie eine abhängige Tabelle von T oder eine untergeordnete Tabelle einer abhängigen Tabelle von T ist.

Abhängige Zeile
Eine Zeile, die mindestens eine übergeordnete Zeile hat.

Untergeordnete Zeile
Eine Zeile ist eine untergeordnete Zeile von Zeile Z, wenn sie eine abhängige Zeile von Z oder eine untergeordnete Zeile einer abhängigen Zeile von Z ist.

Referenzieller Zyklus
Eine Gruppe referenzieller Integritätsbedingungen, in der jede Tabelle in der Gruppe eine untergeordnete Tabelle zu sich selbst ist.

Auf sich selbst verweisende Tabelle
Eine Tabelle, die in der gleichen referenziellen Integritätsbedingung sowohl übergeordnete als auch abhängige Tabelle ist. Die entsprechende Integritätsbedingung wird als *auf sich selbst verweisende Integritätsbedingung* bezeichnet.

Auf sich selbst verweisende Zeile
Eine Zeile, die eine übergeordnete Zeile zu sich selbst ist.

Einfügeregel

Die Einfügeregel einer referenziellen Integritätsbedingung legt fest, dass ein einzufügender Nicht-Nullwert des Fremdschlüssels mit einem Wert des übergeordneten Schlüssels der übergeordneten Tabelle übereinstimmen muss. Der Wert eines zusammengesetzten Fremdschlüssels ist NULL, wenn eine Komponente des Werts NULL ist. Diese Regel gilt implizit, wenn ein Fremdschlüssel definiert wird.

Aktualisierungsregel

Die Aktualisierungsregel einer referenziellen Integritätsbedingung wird bei der Definition der referenziellen Integritätsbedingung angegeben. Angegeben werden kann NO ACTION oder RESTRICT. Die Aktualisierungsregel wird angewendet, wenn eine Zeile der übergeordneten Tabelle oder eine Zeile der abhängigen Tabelle aktualisiert wird.

Für eine übergeordnete Zeile gelten folgende Regeln, wenn ein Wert in einer Spalte des übergeordneten Schlüssels aktualisiert wird:

- Wenn irgendeine Zeile in der abhängigen Tabelle mit dem Ursprungswert des Schlüssels übereinstimmt, wird die UPDATE-Operation zurückgewiesen, wenn als Aktualisierungsregel RESTRICT definiert ist.
- Wenn irgendeine Zeile der abhängigen Tabelle nach Ausführung der UPDATE-Anweisung (ausgenommen Nachauslöser) keinen entsprechenden übergeordneten Schlüssel hat, wird die Aktualisierung zurückgewiesen, wenn als Aktualisierungsregel NO ACTION definiert ist.

Für eine abhängige Zeile gilt die Aktualisierungsregel NO ACTION implizit, wenn ein Fremdschlüssel angegeben wird. NO ACTION bedeutet, dass ein Aktualisierungswert eines Fremdschlüssels, der nicht NULL ist, mit einem Wert des übergeordneten Schlüssels der übergeordneten Tabelle übereinstimmen muss, wenn die UPDATE-Anweisung ausgeführt ist.

Der Wert eines zusammengesetzten Fremdschlüssels ist NULL, wenn eine Komponente des Werts NULL ist.

Löschregel

Die Löschregel einer referenziellen Integritätsbedingung wird bei der Definition der referenziellen Integritätsbedingung angegeben. Angegeben werden kann NO ACTION, RESTRICT, CASCADE oder SET NULL. SET NULL kann nur angegeben werden, wenn eine Spalte des Fremdschlüssels Nullwerte zulässt.

Die Löschregel einer referenziellen Integritätsbedingung wird angewendet, wenn eine Zeile der übergeordneten Tabelle gelöscht wird. Genauer gesagt, die Regel wird angewendet, wenn eine Zeile der übergeordneten Tabelle das Objekt einer Löschoperation bzw. einer weitergegebenen Löschoperation (weiter unten definiert) ist und die Zeile abhängige Zeilen in der abhängigen Tabelle der referenziellen Integritätsbedingung hat. Betrachten Sie ein Beispiel, in dem P die übergeordnete Tabelle, D die abhängige Tabelle und p die übergeordnete Zeile sind, die das Objekt einer Löschoperation bzw. einer weitergegebenen Löschoperation ist. Die Löschregel funktioniert wie folgt:

- Bei RESTRICT oder NO ACTION tritt ein Fehler auf und keine Zeilen werden gelöscht.
- Bei CASCADE wird die Löschoperation an die abhängigen Zeilen von p in Tabelle D weitergegeben.
- Bei SET NULL wird jede Spalte mit optionaler Dateneingabe (NULLABLE) des Fremdschlüssels jeder abhängigen Zeile von p in Tabelle D auf den Wert NULL gesetzt.

Jede referenzielle Integritätsbedingung, in der eine Tabelle eine übergeordnete Tabelle ist, besitzt eine eigene Löschregel, und alle geltenden Löschregeln zusammen legen das Ergebnis einer Löschoperation fest. Daher kann eine Zeile nicht gelöscht werden, wenn sie abhängige Zeilen in einer referenziellen Integritätsbedingung mit der Löschregel RESTRICT oder NO ACTION hat bzw. wenn die Löschung sich auf untergeordnete Zeilen auswirken würde, die in einer referenziellen Integritätsbedingung mit der Löschregel RESTRICT oder NO ACTION abhängige Zeilen sind.

An der Löschung einer Zeile aus der übergeordneten Tabelle P sind in folgenden Fällen weitere Tabellen beteiligt, und die Löschung kann sich auf Zeilen dieser Tabelle auswirken:

- Wenn Tabelle D eine abhängige Tabelle von P ist und die Löschregel RESTRICT oder NO ACTION definiert ist, dann ist Tabelle D an dieser Operation beteiligt, jedoch wirkt sich die Operation nicht auf sie aus.
- Wenn Tabelle D eine abhängige Tabelle von P ist und die Löschregel SET NULL definiert ist, dann ist D an der Operation beteiligt und Zeilen von D können während der Operation aktualisiert werden.
- Wenn Tabelle D eine abhängige Tabelle von P ist und die Löschregel CASCADE definiert ist, dann ist D an der Operation beteiligt und Zeilen von D können während der Operation gelöscht werden.

  Wenn Zeilen der Tabelle D gelöscht werden, wird die Löschoperation an Tabelle P als an Tabelle D weitergegeben bezeichnet. Wenn D gleichzeitig eine übergeordnete Tabelle ist, werden die Aktionen dieser Liste wiederum auf die abhängigen Tabellen von D angewendet.

Jede Tabelle, die an einer Löschoperation von Tabelle P beteiligt sein kann, wird als durch *übergreifendes Löschen* mit P verbunden bezeichnet. Das heißt, eine Tabelle ist mit Tabelle P durch übergreifendes Löschen verbunden, wenn sie eine abhängige Tabelle von P oder einer Tabelle ist, an die Löschoperationen von P weitergegeben werden.

Die folgenden Einschränkungen gelten für die Abhängigkeiten zwischen Tabellen, die durch übergreifendes Löschen miteinander verbunden sind:

- Wenn eine Tabelle durch übergreifendes Löschen in einem referenziellen Zyklus von mehr als einer Tabelle mit sich selbst verbunden ist, darf der Zyklus keine Löschregel RESTRICT oder SET NULL enthalten.
- Eine Tabelle darf nicht gleichzeitig eine abhängige Tabelle in einer CASCADE-Abhängigkeit (auf sich selbst oder auf eine andere Tabelle verweisend) sein und eine auf sich selbst verweisende Beziehung mit einer Löschregel RESTRICT oder SET NULL haben.
- Wenn eine Tabelle durch übergreifendes Löschen mit einer anderen Tabelle über mehrere Abhängigkeiten verbunden ist, wobei diese Abhängigkeiten überlappende Fremdschlüssel haben, müssen diese Abhängigkeiten die gleiche Löschregel haben und keine dieser Löschregeln kann SET NULL sein.
- Wenn eine Tabelle durch übergreifendes Löschen mit einer anderen Tabelle über mehrere Abhängigkeiten verbunden ist, wobei eine der Beziehungen mit der Löschregel SET NULL angegeben ist, darf die Fremdschlüsseldefinition dieser Abhängigkeit keinen Partitionierungsschlüssel oder eine MDC-Schlüsselspalte enthalten.
- Wenn zwei Tabellen durch übergreifendes Löschen mit derselben Tabelle über CASCADE-Abhängigkeiten verbunden sind, dürfen die beiden Tabellen nicht durch ein übergreifendes Löschen miteinander verbunden werden, bei dem die Pfade der durch übergreifendes Löschen definierten Abhängigkeiten mit der Löschregel RESTRICT oder SET NULL enden.

Prüfungen auf Integritätsbedingungen in Tabellen

Eine *Prüfung auf Integritätsbedingung in einer Tabelle* ist eine Regel, mit der die zulässigen Werte in einer oder mehreren Spalten jeder Zeile einer Tabelle angegeben werden. Diese Prüfung auf Integritätsbedingung ist optional und kann mit der Anweisung CREATE TABLE oder ALTER TABLE definiert werden. Die Angabe von Prüfungen auf Integritätsbedingungen in einer Tabelle erfolgt durch eine eingeschränkte Form einer Suchbedingung. Eine der Einschränkungen besteht darin, dass ein Spaltenname in einer Prüfung auf Integritätsbedingung in Tabelle T eine Spalte der Tabelle T angeben muss.

Eine Tabelle kann eine beliebige Anzahl von Prüfungen auf Integritätsbedingungen haben. Eine Prüfung auf Integritätsbedingung ausgeführt, indem die entsprechende Suchbedingung auf jede Zeile, die eingefügt oder aktualisiert wird, angewendet wird. Wenn die Suchbedingung für jede Zeile falsch ist, tritt ein Fehler auf.

Wenn eine oder mehrere Prüfungen auf Integritätsbedingungen in der Anweisung ALTER TABLE für eine Tabelle mit vorhandenen Daten definiert wird, werden die vorhandenen Daten gegen die neue Bedingung geprüft, bevor die Ausführung der Anweisung ALTER TABLE beendet wird. In diesem Fall kann die Tabelle mit der Anweisung SET INTEGRITY in den Status *Überprüfung anstehend* versetzt werden, wodurch die Anweisung ALTER TABLE fortfahren kann, ohne die vorhandenen Daten zu prüfen.

Informative Integritätsbedingungen

Eine *informative Integritätsbedingung* ist eine Regel, die vom SQL-Compiler zur Verbesserung des Zugriffspfads auf Daten verwendet werden kann. Informative Integritätsbedingungen werden nicht durch den Datenbankmanager umgesetzt und dienen nicht zur weiteren Überprüfung von Daten, sondern zur Verbesserung der Abfrageleistung.

Verwenden Sie die Anweisung CREATE TABLE oder ALTER TABLE, um eine referenzielle Integritätsbedingung oder eine Prüfung auf Integritätsbedingung in Tabellen zu definieren, indem Sie Attribute für die Integritätsbedingung angeben, die festlegen, ob der Datenbankmanager die Integritätsbedingung umsetzen soll und ob die Integritätsbedingung zur Abfrageoptimierung herangezogen werden soll oder nicht.

Zugehörige Referenzen:

- „SET INTEGRITY statement" in *SQL Reference, Volume 2*
- „Interaction of triggers and constraints" in *SQL Reference, Volume 1*

Auslöser

Ein *Auslöser* definiert eine Reihe von Aktionen, die in Reaktion auf eine Einfüge-, Aktualisierungs- oder Löschoperation für eine angegebene Tabelle ausgeführt werden. Wenn eine solche SQL-Operation ausgeführt wird, wird der Auslöser als *aktiviert* bezeichnet.

Auslöser sind optional und werden mit Hilfe der Anweisung CREATE TRIGGER definiert.

Auslöser können zusammen mit referenziellen Integritätsbedingungen und Prüfungen auf Integritätsbedingungen in Tabellen zur Implementierung von Datenintegritätsregeln eingesetzt werden. Darüber hinaus können Auslöser auch genutzt werden, um Aktualisierungen an anderen Tabellen zu bewirken, automatisch Werte für eingefügte oder aktualisierte Zeilen zu generieren bzw. umzuwandeln oder Funktion für solche Operationen wie das Absetzen von Alerts aufzurufen.

Auslöser bilden einen nützlichen Mechanismus zur Definition und Implementierung von *Übergangsregeln* für den Geschäftsbetrieb, bei denen es sich um Regeln handelt, die für verschiedene Status der Daten gelten (z. B. ein Gehalt, dass nicht um mehr als zehn Prozent angehoben werden kann).

Mit Hilfe von Auslösern lässt sich die Logik, durch die Geschäftsregeln implementiert werden, außerhalb der Datenbank verwalten. Das heißt, dass nicht die Anwendungen für die Implementierung dieser Regeln zuständig sind. Eine zentralisierte Logik, die für alle Tabellen implementiert wird, ermöglicht eine einfachere Pflege, da keine Änderungen an Anwendungsprogrammen erforderlich werden, wenn sich die Logik ändert.

Zur Erstellung eines Auslösers sind folgende Angaben erforderlich:

- Die *Subjekttabelle* gibt die Tabelle an, für die der Auslöser definiert wird.
- Das *Auslöseereignis* definiert eine bestimmte SQL-Operation, die die Subjekttabelle modifiziert. Bei dem Ereignis kann es sich um eine Einfüge-, Aktualisierungs- oder Löschoperation handeln.
- Die *Aktivierungszeit des Auslösers* gibt an, ob der Auslöser vor oder nach dem Eintreten des Auslöserereignisses zu aktivieren ist.

Die Anweisung, welche die Aktivierung eines Auslösers bewirkt, enthält eine *Gruppe der betroffenen Zeilen*. Dies sind die Zeilen der Subjekttabelle, die eingefügt, aktualisiert oder gelöscht werden. Die *Auslösergranularität* gibt an, ob die Aktionen des Auslösers einmal für die Anweisung oder einmal für jede der betroffenen Zeilen ausgeführt werden.

Die *ausgelöste Aktion* besteht aus einer optionalen Suchbedingung und einer Gruppe von SQL-Anweisungen, die ausgeführt werden, wenn der Auslöser aktiviert wird. Die SQL-Anweisungen werden nur ausgeführt, wenn die Suchbedingung ein wahres Ergebnis liefert. Wenn die Ausführungszeit des Auslösers vor dem Auslöseereignis liegt, können die ausgelösten Aktionen Anweisungen enthalten, die Übergangsvariablen festlegen oder SQLSTATE-Werte signalisieren. Liegt die Ausführungszeit des Auslösers nach dem Auslöseereignis, können die ausgelösten Aktionen Anweisungen enthalten, die auswählen (SELECT), aktualisieren (UPDATE), Löschen (DELETE) oder SQLSTATE-Werte signalisieren.

Die ausgelöste Aktion kann sich mit Hilfe von *Übergangsvariablen* auf die Werte in der Gruppe der betroffenen Zeilen beziehen. Übergangsvariablen verwenden die Namen der Spalten in der Subjekttabelle, die durch einen angegebenen Namen qualifiziert werden, der zu erkennen gibt, ob es sich um einen Verweis auf den alten Wert (vor der Aktualisierung) oder den neuen Wert (nach der Aktualisierung) handelt. Der neue Wert kann mit Hilfe der Anweisung SET Variable in Vorauslösern oder Einfüge- und Aktualisierungsauslösern ebenfalls geändert werden.

Eine weitere Methode zur Bezugnahme auf die Werte in der Gruppe der betroffenen Zeilen besteht in der Verwendung von *Übergangstabellen*. Übergangstabellen können ebenfalls die Namen von Spalten in der Subjekttabelle verwenden, stellen jedoch einen Namen bereit, über den die gesamte Gruppe der betroffenen Zeilen wie eine Tabelle behandelt werden kann. Übergangstabellen können nur in Nachauslösern verwendet werden, wobei für alte und neue Werte getrennte Übergangstabellen definiert werden können.

Es können mehrere Auslöser angegeben werden, um Tabellen, Ereignisse oder Aktivierungszeiten zu kombinieren. Die Reihenfolge, in der die Auslöser aktiviert werden, entspricht der Reihenfolge, in der sie erstellt werden. Das heißt, der zuletzt erstellte Auslöser wird auch zuletzt aktiviert.

Die Aktivierung eines Auslösers kann zu einem *Hintereinanderschalten von Auslösern* führen. Diese Bezeichnung bezieht sich auf das Ergebnis der Aktivierung eines Auslösers, der SQL-Anweisungen ausführt, die wiederum die Aktivierung anderer

Auslöser oder sogar erneut desselben Auslösers bewirken. Die ausgelösten Aktionen können außerdem Aktualisierungen bewirken, die sich aus der Anwendung von Regeln der referenziellen Integrität für Löschungen ergeben, die wiederum die Aktivierung weiterer Auslöser nach sich ziehen können. Durch die Hintereinanderschaltung von Auslösern kann eine Kette von Auslösern und Löschregeln der referenziellen Integrität aktiviert werden, die einen erheblichen Umfang an Änderungen an der Datenbank als Folge einer einzigen INSERT-, UPDATE- oder DELETE-Anweisung bewirken.

Zugehörige Referenzen:

- „Interaction of triggers and constraints" in *SQL Reference, Volume 1*

Zusätzliche Überlegungen zum Datenbankentwurf

Beim Entwurf einer Datenbank muss vorausgeplant werden, auf welche Tabellen Benutzer Zugriff haben sollten. Der Zugriff auf Tabellen wird mit Hilfe von Berechtigungen erteilt oder entzogen. Die höchste Ebene der Berechtigung ist die Berechtigung zur Systemverwaltung (SYSADM). Ein Benutzer mit der Berechtigung SYSADM kann andere Berechtigungen, einschließlich der Berechtigung für den Datenbankadministrator (DBADM), erteilen.

Zur Erfassung von *Prüfdaten* muss eventuell jede Aktualisierung an den Daten über einen bestimmten Zeitraum hinweg aufgezeichnet werden. Sie können zum Beispiel eine Prüfdatentabelle jedes Mal dann aktualisieren, wenn das Gehalt eines Mitarbeiters geändert wird. Aktualisierungen dieser Tabelle könnten automatisch durchgeführt werden, wenn ein entsprechender Auslöser (Trigger) definiert wird. Prüfaktivitäten können außerdem über die Prüffunktion von DB2® Universal Database (DB2 UDB) ausgeführt werden.

Aus Gründen der Leistung kann es sinnvoll sein, nur auf eine ausgewählte Menge von Daten zuzugreifen, während die Basisdaten als *Protokoll (History)* verwaltet werden. Sie sollten in Ihren Datenbankentwurf die Anforderungen zur Verwaltung solcher Protokolldaten aufnehmen, z. B. die Anzahl der Monate oder Jahre, die Daten verfügbar sein müssen, bevor sie gelöscht werden können.

Es kann außerdem sinnvoll sein, *Übersichtsdaten* zu erstellen. Zum Beispiel könnten Sie über eine Tabelle verfügen, die Ihre gesamten Mitarbeiterdaten enthält. Angenommen, Sie möchten diese Daten nach Unternehmensbereichen oder Abteilungen in separate Tabellen aufteilen. In diesem Fall wäre es hilfreich, gespeicherte Abfragetabellen für die einzelnen Unternehmensbereiche oder Abteilungen anzulegen, die auf den Daten der Originaltabelle basieren.

In Ihrem Datenbankentwurf sollten auch die *Sicherheitsaspekte* berücksichtigt werden. Sie können zum Beispiel den Entwurf so gestalten, dass der Zugriff von Benutzern auf bestimmte Arten von Daten über Sicherheitstabellen unterstützt wird. Sie können Zugriffsebenen für verschiedene Arten von Daten und verschiedene Benutzer mit Zugriffberechtigung für diese Daten definieren. Vertrauliche Daten, wie Personal- und Gehaltsdaten, könnten strikten Sicherheitsbeschränkungen unterliegen.

Sie können Tabellen erstellen, denen ein *strukturierter Typ* zugeordnet ist. Mit solchen typisierten Tabellen können Sie eine hierarchische Struktur mit einer definierten Abhängigkeitsbeziehung zwischen diesen Tabellen erstellen, die als *Typhierarchie* bezeichnet wird. Eine Typhierarchie besteht aus einem Stammtyp sowie aus übergeordneten und untergeordneten Datentypen.

Die Darstellung eines *Referenztyps* wird beim Erstellen des Stammtyps einer Typhierarchie definiert. Das Ziel einer Referenz ist immer eine Zeile in einer typisierten Tabelle oder Sicht.

Für die Arbeit in einer HADR-Umgebung (High Availability Disaster Recovery) gelten verschiedene Empfehlungen:

- Zwei Exemplare der HADR-Umgebung sollten in Hard- und Software identisch sein. Dies bedeutet:
  - Die Hosts für die HADR-Primär- und Bereitschaftsdatenbank sollten identisch sein.
  - Die Betriebssysteme auf dem Primär- und dem Bereitschaftssystem sollten einschließlich der Programmkorrekturen dieselbe Version aufweisen. (Während eines Updates ist dies vielleicht nicht möglich. Allerdings sollte der Zeitraum, über den die Exemplare nicht die gleiche Stufe haben, möglichst kurz gehalten werden, um Schwierigkeiten, die sich aus diesen Unterschieden ergeben, zu begrenzen. Solche Schwierigkeiten könnten zum Beispiel ein Verlust der Unterstützung für neue Funktionen während einer Funktionsübernahme sowie Probleme im Normalbetrieb ohne Funktionsübernahme sein.)
  - Eine TCP/IP-Schnittstelle muss zwischen den beiden HADR-Exemplaren zur Verfügung stehen.
  - Ein Hochgeschwindigkeitsnetzwerk mit hoher Kapazität wird empfohlen.
- Einige durch DB2 UDB bedingte Anforderungen sollten beachtet werden.
  - Das Primär- und das Bereitschaftssystem sollten mit identischen Datenbankreleases arbeiten.
  - Die DB2 UDB-Software des Primär- und des Bereitschaftssystem müssen die gleiche Bitgröße besitzen. (Das heißt, beide sollten entweder 32-Bit-Systeme oder 64-Bit-Systeme sein.)
  - Die Tabellenbereiche und die entsprechenden Behälter sollten für die Primär- und die Bereitschaftsdatenbank identisch sein. Zu den Merkmalen, die für Tabellenbereiche symmetrisch sein müssen, gehören u. a. die folgenden: der Tabellenbereichstyp (DMS oder SMS), die Tabellenbereichsgröße, die Behälterpfade, die Behältergrößen und der Behälterdateityp (unformatierte Einheit oder Dateisystem). Relative Behälterpfade können verwendet werden, wobei der relative Pfad in jedem Exemplar der gleiche sein muss. Sie können gleichen oder verschiedenen absoluten Pfaden zugeordnet sein.
  - Die Primär- und die Bereitschaftsdatenbank brauchen nicht denselben Datenbankpfad (wie bei der Verwendung des Befehls CREATE DATABASE deklariert) zu besitzen.
  - Pufferpooloperationen auf dem Primärsystem werden auf dem Bereitschaftssystem nachvollzogen, so dass einsichtig ist, warum gleiche Speichergrößen in der Primär- und der Bereitschaftsdatenbank eine wichtige Rolle spielen.

Kapitel 5. Physischer Datenbankentwurf

Nachdem Sie den logischen Datenbankentwurf entwickelt haben, gibt es eine Reihe von Aspekten, die hinsichtlich der physischen Umgebung, in der die Datenbank und die Tabellen angelegt werden, zu berücksichtigen sind. Hierzu gehören Kenntnisse bezüglich der Dateien, die für die Unterstützung und Verwaltung Ihrer Datenbank erstellt werden, eine Vorstellung davon, wie viel Speicherbereich zum Speichern Ihrer Daten erforderlich ist, die Bestimmung, wie die zum Speichern Ihrer Daten benötigten Tabellenbereiche verwendet werden sollen, sowie die Struktur der Tabellen, die zur Aufnahme der Daten verwendet werden sollen.

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_exemplar>/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" auf Seite 125
- „Überlegungen zu DMS-Einheiten" in *Systemverwaltung: Optimierung*
- „Vom Betriebssystem verwaltete Tabellenbereiche" in *Systemverwaltung: Optimierung*
- „Vom Datenbankmanager verwaltete Tabellenbereiche" in *Systemverwaltung: Optimierung*
- „Darstellung der Zuordnung von DMS-Tabellenbereichsadressen" in *Systemverwaltung: Optimierung*
- „Datei des Wiederherstellungsprotokolls" in *Datenwiederherstellung und hohe Verfügbarkeit Handbuch und Referenz*

Zugehörige Referenzen:

- „CREATE DATABASE Command" in *Command Reference*

Speicherbedarf für Datenbankobjekte

Das Abschätzen der Größe von Datenbankobjekten ist ein ungenaues Unterfangen. Der Systemaufwand, der durch Datenträgerfragmentierung, freien Speicherbereich und die Verwendung von Spalten variabler Länge bedingt ist, macht eine Abschätzung schwierig, weil es eine große Bandbreite an Möglichkeiten für Spaltentypen und Zeilenlängen gibt. Erstellen Sie nach dem Abschätzen der Größe Ihrer Datenbank eine Testdatenbank, und füllen Sie sie mit repräsentativen Daten.

Über die Steuerzentrale können Sie auf eine Reihe von Dienstprogrammen zugreifen, die Sie bei der Ermittlung der Größenanforderungen verschiedener Datenbankobjekte unterstützen:

- Sie können ein Objekt auswählen und das Dienstprogramm "Größe schätzen" verwenden. Mit diesem Dienstprogramm können Sie die aktuelle Größe eines vorhandenen Objekts, wie z. B. einer Tabelle, ermitteln. Sie können dann das Objekt ändern, und das Dienstprogramm berechnet neue Schätzwerte für das Objekt. Dieses Dienstprogramm hilft bei der Abschätzung des ungefähren Speicherbedarfs unter Berücksichtigung zukünftiger Zuwächse. Es bietet mehr als einen einzelnen Schätzwert für die Größe des Objekts. Es gibt auch mögliche Größenbereiche für das Objekt an: die minimale Größe, die auf den aktuellen Werten beruht, und die mögliche Maximalgröße.
- Sie können die Abhängigkeiten zwischen Objekten bestimmen, indem Sie das Fenster "Zugehörige anzeigen" verwenden.
- Sie können ein beliebiges Datenbankobjekt im Exemplar auswählen und "DDL generieren" anfordern. Diese Funktion verwendet das Dienstprogramm **db2look**, um Datendefinitionsanweisungen für die Datenbank zu generieren.

In allen diesen Fällen steht Ihnen entweder der Knopf "SQL anzeigen" oder der Knopf "Befehl anzeigen" zur Verfügung. Sie können die generierten SQL-Anweisungen oder Befehle auch in Prozedurdateien sichern, um diese später zu verwenden. In allen diesen Dienstprogrammen werden Sie durch eine Onlinehilfefunktion unterstützt.

Sie sollten diese Einrichtungen bei der Planung Ihrer physischen Datenbankanforderungen berücksichtigen.

Bei der Abschätzung der Größe einer Datenbank müssen die Auswirkungen der folgenden Elemente berücksichtigt werden:

- Systemkatalogtabellen
- Benutzertabellendaten
- Langfelddaten
- LOB-Daten (Large Object)
- Indexspeicherbereich
- Speicherbereich für Protokolldateien
- Temporärer Arbeitsspeicherbereich

Der Platzbedarf folgender Elemente wird nicht behandelt:

- Datei für das lokale Datenbankverzeichnis
- Datei für das Systemdatenbankverzeichnis
- Bei der Dateiverwaltung entstehender Systemaufwand, den das Betriebssystem benötigt, zum Beispiel:
  - Dateiblockgröße
  - Speicherbereich für die Verzeichnissteuerung

Zugehörige Konzepte:

- „Speicherbedarf für Systemkatalogtabellen" auf Seite 83
- „Speicherbedarf für Benutzertabellendaten" auf Seite 83
- „Speicherbedarf für Langfelddaten" auf Seite 85
- „Speicherbedarf für LOB-Daten" auf Seite 86
- „Speicherbedarf für Indizes" auf Seite 87

- „Speicherbedarf für Protokolldateien" auf Seite 89
- „Speicherbedarf für temporäre Tabellen" auf Seite 91

Zugehörige Referenzen:

- „db2look - DB2 Statistics and DDL Extraction Tool Command" in *Command Reference*

Speicherbedarf für Systemkatalogtabellen

Systemkatalogtabellen werden erstellt, wenn eine Datenbank erstellt wird. Der Umfang dieser Systemtabellen nimmt in dem Maße zu, wie der Datenbank Datenbankobjekte und Zugriffsrechte hinzugefügt werden. Zu Beginn belegen diese Tabellen einen Plattenspeicherplatz von ungefähr 3,5 MB.

Die Größe des Speicherbereichs, der für die Katalogtabellen zugeordnet wird, hängt von der Art des Tabellenbereichs und der Größe des durch EXTENTSIZE definierten Bereichs des Tabellenbereichs mit den Katalogtabellen ab. Wenn z. B. ein DMS-Tabellenbereich mit einem EXTENTSIZE-Wert von 32 verwendet wird, wird dem Katalogtabellenbereich anfangs ein Speicherbereich von 20 MB zugeordnet.

Anmerkung: Für Datenbanken mit mehreren Partitionen befinden sich die Katalogtabellen nur in der Partition, von der aus der Befehl CREATE DATABASE abgesetzt wurde. Plattenspeicherplatz für die Katalogtabellen ist nur für diese Partition erforderlich.

Zugehörige Konzepte:

- „Speicherbedarf für Datenbankobjekte" auf Seite 81
- „Definition von Systemkatalogtabellen" in *Systemverwaltung: Implementierung*

Speicherbedarf für Benutzertabellendaten

Standardmäßig werden die Tabellendaten auf Seiten von jeweils 4 KB gespeichert. Jede Seite enthält (unabhängig von der Seitengröße) 68 Byte Systemaufwand für den Datenbankmanager. Dadurch bleiben 4028 Byte für die Benutzerdaten (oder Zeilen), obwohl keine Zeile auf einer 4-KB-Seite die Länge von 4005 Byte überschreiten kann. Eine Zeile kann sich *nicht* über mehrere Seiten erstrecken. Wenn Sie Seiten mit jeweils 4 KB verwenden, sind maximal 500 Spalten möglich.

Tabellendatenseiten enthalten *keine* Daten für Spalten mit den Datentypen LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB oder DBCLOB. Die Zeilen einer Tabellendatenseite enthalten jedoch einen Deskriptor für diese Spalten.

Zeilen werden normalerweise an der ersten passenden Stelle in einer regulären Tabelle eingefügt. Die Datei wird nach dem ersten verfügbaren Speicherbereich durchsucht (unter Verwendung einer Liste der freien Speicherbereiche), der groß genug ist, um die neue Zeile aufzunehmen. Das Aktualisieren einer vorhandenen Zeile erfolgt an der ursprünglichen Stelle, es sei denn, der freie Bereich der 4-KB-Seite reicht nicht aus, um die aktualisierte Zeile aufzunehmen. In diesem Fall wird an der Position, an der sich die Originalzeile befand, ein Datensatz erstellt, der auf die neue Speicherposition der aktualisierten Zeile in der Tabellendatei verweist.

Bei Verwendung der Anweisung ALTER TABLE APPEND ON werden Daten immer angehängt und es werden keine Informationen zum freien Speicherbereich der Datenseiten aufgezeichnet.

Wenn die Tabellen einen definierten Clusterindex besitzt, versucht DB2® Universal Database (DB2 UDB), die Daten entsprechend der Schlüsselreihenfolge dieses Clusterindex physisch (zu Clustern) zusammenzufassen. Wenn eine Zeile in die Tabelle eingefügt wird, sucht DB2 UDB zuerst den entsprechenden Schlüsselwert im Clusterindex. Wird der Schlüsselwert gefunden, versucht DB2 UDB den Datensatz auf der durch den Schlüssel angegebenen Datenseite einzufügen. Wird der Schlüsselwert nicht gefunden, wird der nächst höhere Schlüsselwert verwendet, so dass der Datensatz auf der Seite eingefügt wird, die Datensätze mit dem nächst höheren Schlüsselwert enthält. Wenn auf der „Zielseite" nicht genügend freier Speicherbereich zur Verfügung steht, wird die Liste der freien Speicherbereiche zur Suche nach benachbarten Seiten mit freiem Speicherbereich verwendet. Mit der Zeit wird der Speicherbereich auf den Datenseiten vollständig belegt, und Datensätze werden immer weiter von der „Zielseite" entfernt in der Tabelle gespeichert. Schließlich werden die Tabellendaten als Daten ohne Clustering betrachtet. In diesem Fall kann die Clusterreihenfolge durch eine Tabellenreorganisation wiederhergestellt werden.

Wenn es sich bei der Tabelle um eine MDC-Tabelle (mit mehrdimensionalem Clustering) handelt, garantiert DB2 UDB, dass Datensätze immer entlang der Dimension bzw. Dimensionen bzw. entsprechend den Clusterindizes physisch in Clustern angeordnet werden. Wenn eine MDC-Tabelle mit bestimmten Dimensionen definiert wird, wird ein Blockindex für jede der Dimensionen sowie ein zusammengesetzter Blockindex erstellt, der Zellen (eindeutige Kombinationen aus Dimensionswerten) Blöcken zuordnet. Dieser zusammengesetzte Blockindex dient zur Bestimmung, zu welcher Zelle ein bestimmter Datensatz gehört und welche Blöcke oder EXTENTSIZE großen Speicherbereiche in der Tabelle Datensätze enthalten, die zu dieser Zelle gehören. Beim Einfügen von Datensätzen durchsucht DB2 UDB daher den zusammengesetzten Blockindex nach der Liste von Blöcken, die Datensätze mit den gleichen Dimensionswerten enthalten, und beschränkt die Suche nach freiem Speicherplatz auf diese Blöcke. Wenn die Zelle noch nicht vorhanden ist oder wenn in den vorhanden Blöcken der Zelle kein ausreichender Speicherbereich zur Verfügung steht, wird der Zelle ein weiterer Block zugeordnet und der Datensatz in diesen eingefügt. Eine Liste der freien Speicherbereiche wird weiterhin innerhalb von Blöcken verwendet, um rasch verfügbaren Speicherbereich in den Blöcken ausfindig zu machen.

Die Anzahl der 4-KB-Seiten für jede Benutzertabelle in der Datenbank kann nach folgender Berechnung abgeschätzt werden:

```
ABRUNDEN(4028/(durchschnittszeilengröße + 10)) = datensätze_pro_seite
```

Das Ergebnis wird in folgende Berechnung eingefügt:

```
(zahl_der_datensätze/datensätze_pro_seite) * 1,1 = anzahl_der_seiten
```

Dabei ist die durchschnittliche Zeilenlänge die Summe der durchschnittlichen Spaltengrößen und der Faktor ″1,1″ dient zur Kalkulation des Systemaufwands.

Anmerkung: Diese Formel ergibt nur einen Schätzwert. Die Genauigkeit des Schätzwerts nimmt ab, wenn die Datensatzlänge durch Fragmentierung und Überlaufsätze variiert.

Sie können auch Pufferpools oder Tabellenbereiche mit einer Seitengröße von 8 KB, 16 KB oder 32 KB erstellen. Alle Tabellen, die innerhalb eines Tabellen-

bereichs einer bestimmten Größe erstellt wurden, besitzen eine übereinstimmende Seitengröße. Die maximale Größe für ein einzelnes Tabellen- oder Indexobjekt beträgt 512 GB bei einer einer Seitengröße von 32 KB. Wenn Sie Seiten mit einer Größe von 8 KB, 16 KB oder 32 KB verwenden, sind maximal 1012 Spalten möglich. Für eine 4-KB-Seite beträgt die maximale Spaltenanzahl 500. Die maximalen Zeilenlängen variieren ebenfalls je nach Seitengröße:

- Bei einer Seitengröße von 4 KB beträgt die maximale Zeilenlänge 4005 Byte.
- Bei einer Seitengröße von 8 KB beträgt die maximale Zeilenlänge 8101 Byte.
- Bei einer Seitengröße von 16 KB beträgt die maximale Zeilenlänge 16 293 Byte.
- Bei einer Seitengröße von 32 KB beträgt die maximale Zeilenlänge 32 677 Byte.

Durch größere Seiten wird es möglich, die Zahl der Indexstufen in einem Index zu verringern. Wenn Sie mit OLTP-Anwendungen arbeiten, die wahlfreie Lese- und Schreibzugriffe durchführen, ist eine geringere Seitengröße empfehlenswert, weil dadurch weniger Pufferspeicher durch unerwünschte Zeilen belegt wird. Wenn Sie mit DSS-Anwendungen (Decision Support System) arbeiten, die jeweils auf eine große Anzahl aufeinander folgender Zeilen zugreifen, sind größere Seiten empfehlenswert, weil dadurch weniger E/A-Anforderungen erforderlich sind, um eine bestimmte Anzahl Zeilen zu lesen. Eine Ausnahme bildet der Fall, wenn die Zeilengröße kleiner ist als die Seitengröße dividiert durch 255. In diesem Fall wird auf jeder Seite ungenutzter Speicherbereich verschenkt. (Jede Seite kann maximal nur 255 Zeilen aufnehmen.) Zur Verringerung des unbenutzten Speicherbereichs kann eine geringere Seitengröße besser geeignet sein.

Eine Sicherung kann nicht in einer anderen Seitengröße wiederherstellt werden.

Sie können keine IXF-Datendateien importieren, die mehr als 755 Spalten beinhalten.

Deklarierte temporäre Tabellen können nur in einem eigenen Tabellenbereichstyp für temporäre Benutzertabellen erstellt werden. Es gibt keinen Standardtabellenbereich für temporäre Benutzertabellen. Temporäre Tabellen können keine Daten mit LONG-Typen enthalten. Die Tabellen werden implizit gelöscht, wenn eine Anwendung die Verbindung zur Datenbank trennt. Schätzungen über den entsprechenden Platzbedarf sollten dies berücksichtigen.

Zugehörige Konzepte:

- „Speicherbedarf für Datenbankobjekte” auf Seite 81

Speicherbedarf für Langfelddaten

Langfelddaten werden in einem separaten Tabellenobjekt gespeichert, das anders als bei anderen Datentypen strukturiert ist.

Die Daten werden in Bereichen von 32 KB gespeichert, die sich aus Segmenten zusammensetzen, deren Größen sich aus dem Produkt der Zweierpotenzen und 512 Byte errechnen. (Diese Segmente können also aus 512 Byte, 1024 Byte, 2048 Byte usw. bis 32 768 Byte bestehen.)

Langfelddatentypen (LONG VARCHAR oder LONG VARGRAPHIC) werden auf eine Weise gespeichert, die eine problemlose Neuverwendung freien Speicherbereichs ermöglicht. Die Informationen zur Zuordnung und zu freien Speicherbereichen werden in Zuordnungsseiten von jeweils 4 KB gespeichert, die gelegentlich im Objekt erscheinen.

Der Bereich des freien Speicherplatzes in den Objekten ist von der Größe der Langfelddaten sowie davon abhängig, ob diese Größe bei jedem Vorkommen der Daten relativ konstant ist. Bei Dateneinträgen von mehr als 255 Byte kann dieser nicht genutzte Speicherplatz bis zu 50 Prozent der Größe der Langfelddaten ausmachen.

Wenn Zeichendaten kleiner als die Seitengröße sind und sie zusammen mit dem Rest der Daten in den Datensatz passen, sollten anstelle der Datentypen LONG VARCHAR oder LONG VARGRAPHIC die Datentypen CHAR, GRAPHIC, VARCHAR oder VARGRAPHIC verwendet werden.

Zugehörige Konzepte:

- „Speicherbedarf für Datenbankobjekte" auf Seite 81

Speicherbedarf für LOB-Daten

Daten großer Objekte (LOB-Daten) werden in zwei separaten Tabellenobjekten gespeichert, die anders als der Speicher für andere Datentypen strukturiert sind.

Bei der Abschätzung des für LOB-Daten erforderlichen Speicherbereichs müssen Sie die beiden Tabellenobjekte berücksichtigen, die zum Speichern von Daten dieser Datentypen verwendet werden:

- **LOB-Datenobjekte**

  Die Daten werden in Bereichen von 64 MB gespeichert, die sich aus Segmenten zusammensetzen, deren Größen sich aus dem Produkt der Zweierpotenzen und 1024 Byte errechnen. (Diese Segmente können also aus 1024 Byte, 2048 Byte, 4096 Byte usw. bis 64 MB bestehen.)

  Zur Verringerung des für LOB-Daten erforderlichen Plattenspeicherplatzes können Sie den Parameter COMPACT in der Klausel für die *LOB-Optionen* der Anweisungen CREATE TABLE und ALTER TABLE angeben. Die Option COMPACT minimiert die Größe des für die LOB-Daten erforderlichen Speicherplatzes dadurch, dass die LOB-Daten in kleinere Segmente aufgeteilt werden können. Dieser Prozess beinhaltet keine Datenkomprimierung, sondern verwendet lediglich den kleinstmöglichen Speicherbereich bis zur nächsten 1-KB-Grenze. Die Angabe der Option COMPACT kann zu einer geringeren Leistung führen, wenn Daten an LOB-Werte angehängt werden.

  Der Umfang des freien Speicherbereichs innerhalb der LOB-Datenobjekte wird vom Umfang der Aktualisierungs- und Löschaktivitäten sowie von der Größe der eingefügten LOB-Werte beeinflusst.

- **LOB-Zuordnungsobjekte**

  Die Informationen zur Zuordnung und zu freien Speicherbereichen werden in Zuordnungsseiten von jeweils 4 KB gespeichert, die von den eigentlichen Daten getrennt sind. Die Anzahl dieser 4-KB-Seiten ist vom Umfang der Daten (einschließlich des nicht genutzten Speicherbereichs) abhängig, die für die LOB-Daten zugeordnet wurden. Der Systemaufwand errechnet sich wie folgt: eine 4-KB-Seite pro 64 GB plus eine 4-KB-Seite pro 8 MB.

Wenn Zeichendaten kleiner als die Seitengröße sind und sie zusammen mit dem Rest der Daten in den Datensatz passen, sollten anstelle der Datentypen BLOB, CLOB oder DBCLOB die Datentypen CHAR, GRAPHIC, VARCHAR oder VARGRAPHIC verwendet werden.

Zugehörige Konzepte:

- „Speicherbedarf für Datenbankobjekte" auf Seite 81

Zugehörige Referenzen:

- „Large objects (LOBs)" in *SQL Reference, Volume 1*

Speicherbedarf für Indizes

Für jeden Index kann der erforderliche Speicherbereich wie folgt abgeschätzt werden:

```
(durchschnittliche Indexschlüsselgröße + 9) * Anzahl der Zeilen * 2
```

Dabei gilt Folgendes:

- Die „durchschnittliche Indexschlüsselgröße" ist die Bytezahl jeder Spalte im Indexschlüssel. (Verwenden Sie bei der Abschätzung der durchschnittlichen Spaltengröße für VARCHAR- und VARGRAPHIC-Spalten einen Durchschnittswert der aktuellen Datengröße plus zwei Byte. Verwenden Sie nicht die deklarierte Maximalgröße.)
- Der Faktor 2 ist für den Systemaufwand, z. B. für Nichtblattseiten und freien Speicherbereich.

Anmerkungen:

1. Fügen Sie für jede Spalte, die einen Nullwert (NULL) zulässt, ein zusätzliches Byte für den Nullanzeiger hinzu.
2. Für Blockindizes, die intern für MDC-Tabellen (MDC - Multidimensional Clustering) erstellt werden, würde die „Anzahl der Zeilen" durch die „Anzahl der Blöcke" ersetzt.

Bei der Indexerstellung ist temporärer Speicherplatz erforderlich. Der maximal während der Indexerstellung erforderliche temporäre Speicherplatz kann folgendermaßen abgeschätzt werden:

```
(durchschnittliche Indexschlüsselgröße + 9) * Anzahl der Zeilen * 3,2
```

Dabei wird der Faktor 3,2 für den Indexaufwand sowie für den Speicherbereich einkalkuliert, der für während der Indexerstellung anfallende Sortierungen erforderlich ist.

Anmerkung: Für nicht eindeutige Indizes werden zum Speichern doppelter Schlüsseleinträge nur fünf Byte benötigt. Die oben gezeigten Berechnungen gehen von eindeutigen Indizes aus. Der zum Speichern eines Index erforderliche Speicherbereich kann durch die oben gezeigte Formel eventuell zu groß abgeschätzt werden.

Die beiden folgenden Formeln können zum Abschätzen der Anzahl von Blattseiten eines Index verwendet werden. (Die zweite Formel liefert einen etwas genaueren Schätzwert.) Die Genauigkeit dieser Schätzwerte hängt im Wesentlichen davon ab, wie gut die verwendeten Durchschnittswerte die tatsächlichen Daten widerspiegeln.

Anmerkung: Für SMS-Tabellenbereiche beträgt der minimale Speicherbedarf 12 KB. Für DMS-Tabellenbereiche entspricht der minimale Speicherbedarf einem durch den Wert von EXTENTSIZE definierten Speicherbereich.

- Die Durchschnittszahl von Schlüsseln pro Blattseite kann mit folgender Formel grob abgeschätzt werden:

```
(0,9 * (U - (M*2))) * (D + 1)
----------------------------
      K + 7 + (5 * D)
```

  Dabei gilt Folgendes:
  - U, der verwendbare Speicherbereich auf einer Seite, entspricht ungefähr der Seitengröße minus 100. Bei einer Seitengröße von 4096 ist U gleich 3996.
  - M = U / (9 + *minimale Schlüsselgröße*)
  - D = durchschnittliche Anzahl mehrfacher Werte pro Schlüsselwert
  - K = *durchschnittliche Schlüsselgröße*

  Beachten Sie, dass die Werte für *minimale Schlüsselgröße* und *durchschnittliche Schlüsselgröße* ein Byte extra für jeden Teil des Schlüssels haben müssen, der einen Nullwert enthalten kann, und zusätzliche zwei Byte für die Länge jedes Teils des Schlüssels mit variabler Länge.

  Wenn INCLUDE-Spalten vorhanden sind, müssen sie in den Werten für *minimale Schlüsselgröße* und *durchschnittliche Schlüsselgröße* berücksichtigt werden.

  `0,9` kann durch einen beliebigen, mit (100 - pctfree)/100 berechneten Wert ersetzt werden, wenn während der Indexerstellung ein anderer Prozentwert für freien Speicherbereich (pctfree) angegeben wurde als der Standardwert zehn.
- Die Durchschnittszahl von Schlüsseln pro Blattseite kann mit folgender Formel etwas genauer abgeschätzt werden:

```
L = Blattseitenzahl = X / (Durchschnittszahl von Schlüsseln pro Blattseite)
```

  Dabei ist X die Gesamtzahl der Zeilen in der Tabelle.

  Einen Schätzwert für die Originalgröße eines Indexes können Sie wie folgt berechnen:

```
L + 2L/(Durchschnittszahl von Schlüsseln pro Blattseite)) * Seitengröße
```

  Für DMS-Tabellenbereiche addieren Sie die Größen aller Indizes einer Tabelle zusammen, und runden Sie auf ein Vielfaches des Werts für EXTENTSIZE für den Tabellenbereich auf, in dem sich der Index befindet.

  Sie sollten weiteren Speicherbereich für das Anwachsen des Index durch INSERT/UPDATE-Vorgänge bereitstellen, die zur Teilung von Seiten führen können.

  Durch die folgenden Berechnungen erhalten Sie einen genaueren Schätzwert für die Originalgröße des Indexes sowie einen Schätzwert für die Anzahl der Indexstufen. (Dies ist möglicherweise von besonderem Interesse, wenn in der Indexdefinition INCLUDE-Spalten verwendet werden.) Die Durchschnittszahl von Schlüsseln pro Nichtblattseite kann mit folgender Formel grob abgeschätzt werden:

```
(0,9 * (U - (M*2))) * (D + 1)
----------------------------
      K + 13 + (9 * D)
```

  Dabei gilt Folgendes:
  - U, der verwendbare Speicherbereich auf einer Seite, entspricht ungefähr der Seitengröße minus 100. Bei einer Seitengröße von 4096 ist U gleich 3996.
  - D ist die durchschnittliche Anzahl mehrfacher Werte pro Schlüsselwert auf Nichtblattseiten (dieser Wert ist deutlich kleiner als für Blattseiten. Sie können ihn zur Vereinfachung der Berechnung auf 0 setzen).
  - M = U / (9 + *minimale Schlüsselgröße* für Nichtblattseiten)
  - K = *durchschnittliche Schlüsselgröße* für Nichtblattseiten

Die *minimale Schlüsselgröße* und die *durchschnittliche Schlüsselgröße* für Nichtblattseiten stimmen mit den Werten für Blattseiten überein, sofern keine INCLUDE-Spalten vorkommen. INCLUDE-Spalten werden auf Nichtblattseiten nicht gespeichert.

Sie sollten `0,9` nur durch `(100 - pctfree)/100` ersetzen, wenn dieser Wert größer als `0,9` ist, weil auf Nichtblattseiten bei der Indexerstellung maximal 10% Speicherbereich frei bleibt.

Die Zahl der Nichtblattseiten kann mit folgender Formel abgeschätzt werden:

```
if L > 1 then {P++; Z++}
While (Y > 1)
{
   P = P + Y
   Y = Y / N
  Z++
}
```

Dabei gilt Folgendes:
- P ist die Anzahl von Seiten (zunächst 0).
- L ist die Anzahl der Blattseiten.
- N ist die Anzahl von Schlüsseln pro Nichtblattseite.
- Y = L / N
- Z ist die Anzahl der Ebenen in der Indexbaumstruktur (zunächst 1).

Als Gesamtzahl der Seiten ergibt sich:

```
T = (L + P + 2) * 1,0002
```

Die zusätzlichen 0,02% sind für Systemaufwand (einschließlich Speicherzuordnungsseiten).

Der für die Indexerstellung erforderliche Speicherbereich lässt sich folgendermaßen abschätzen:

```
T * Seitengröße
```

Zugehörige Konzepte:

- „Indexes" in *SQL Reference, Volume 1*
- „Speicherbedarf für Datenbankobjekte" auf Seite 81
- „Bereinigung und Pflege von Indizes" in *Systemverwaltung: Optimierung*

Speicherbedarf für Protokolldateien

Sie benötigen 32 KB Speicherplatz für Protokollsteuerdateien.

Außerdem benötigen Sie mindestens ausreichend Speicherplatz für Ihre aktive Protokollkonfiguration, die Sie nach folgender Formel berechnen können:

```
(logprimary + logsecond) * (logfilsiz + 2 ) * 4096
```

Dabei gilt Folgendes:

- *logprimary* ist die Anzahl der primären Protokolldateien, die in der Konfigurationsdatei der Datenbank definiert sind.
- *logsecond* ist die Anzahl der sekundären Protokolldateien, die in der Konfigurationsdatei der Datenbank definiert sind. In dieser Formel kann *logsecond* nicht auf den Wert *-1* gesetzt werden. (Wenn *logsecond* auf den Wert -1 gesetzt wird, fordern Sie einen unbegrenzten Speicherplatz für aktive Protokolldateien an.)

- *logfilsiz* ist die Anzahl der Seiten in jeder Protokolldatei, die in der Konfigurationsdatei der Datenbank definiert sind.
- 2 ist die Anzahl der Kopfseiten, die für jede Protokolldatei erforderlich ist.
- 4096 ist die Anzahl der Byte einer Seite.

Wenn für die Datenbank die Umlaufprotokollierung aktiviert ist, liefert das Ergebnis dieser Formel einen ausreichenden Betrag für die Größe des benötigten Speicherplatzes.

Wenn für die Datenbank die aktualisierende Wiederherstellung aktiviert ist, sollten spezielle Speicheranforderungen für Protokolle berücksichtigt werden:

- Bei aktiviertem Konfigurationsparameter *logretain* werden die Protokolldateien im Protokollpfadverzeichnis archiviert. Der Online-Plattenspeicherplatz wird schließlich belegt, sofern Sie die Protokolldateien nicht an eine andere Speicherposition verschieben.
- Bei aktiviertem Konfigurationsparameter *userexit* verschiebt ein Benutzer-Exit-Programm die archivierten Protokolldateien an eine andere Speicherposition. Zusätzlicher Speicherbereich ist weiterhin erforderlich für folgende Dateien:
  - Online archivierte Protokolle, die darauf warten, vom Benutzer-Exit-Programm verschoben zu werden
  - Neue Protokolle, die für künftige Zwecke formatiert werden

Wenn für die Datenbank eine unendliche Protokollierung aktiviert ist (d. h., wenn Sie den Parameter *logsecond* auf den Wert *-1* setzen), muss der Konfigurationsparameter *userexit* aktiviert werden, so dass die gleichen Speicherbedarfsüberlegungen gelten. DB2® Universal Database (DB2 UDB) behält mindestens die durch den Parameter *logprimary* definierte Anzahl aktiver Protokolldateien im Protokollpfad, so dass Sie in der obigen Formel für *logsecond* nicht den Wert -1 einsetzen sollten. Stellen Sie sicher, dass Sie zusätzlichen Speicherplatz bereitstellen, um der durch die Archivierung von Protokolldateien verursachten Verzögerung Rechnung zu tragen.

Wenn Sie den Protokollpfad spiegeln, müssen Sie den Schätzwert für den Protokolldateispeicherbedarf verdoppeln.

Zugehörige Konzepte:

- „Speicherbedarf für Datenbankobjekte" auf Seite 81
- „Wiederherstellungsprotokolle" in *Datenwiederherstellung und hohe Verfügbarkeit Handbuch und Referenz*
- „Spiegeln von Protokollen" in *Datenwiederherstellung und hohe Verfügbarkeit Handbuch und Referenz*

Zugehörige Referenzen:

- „logfilsiz - Protokolldateigröße" in *Systemverwaltung: Optimierung*
- „logprimary - Anzahl primärer Protokolldateien" in *Systemverwaltung: Optimierung*
- „logsecond - Anzahl sekundärer Protokolldateien" in *Systemverwaltung: Optimierung*
- „mirrorlogpath - Pfad für Protokollspiegelung" in *Systemverwaltung: Optimierung*

Speicherbedarf für temporäre Tabellen

Einige SQL-Anweisungen benötigen für die Verarbeitung temporäre Tabellen (z. B. eine Arbeitsdatei für Sortieroperationen, die nicht im Speicher vorgenommen werden können). Diese temporären Tabellen benötigen Plattenspeicherplatz. Die Größe des erforderlichen Speichers hängt von der Größe, der Anzahl und der Art der Abfragen sowie von der Größe der Ergebnistabellen ab. Da Ihre Arbeitsumgebung einzigartig ist, lässt sich der Platzbedarf für temporäre Tabellen nur sehr schwer schätzen. Zum Beispiel kann es aufgrund der längeren Lebensdauer verschiedener temporärer Systemdateien so scheinen, als ob mehr Speicherplatz für temporäre Systemtabellen zugeordnet wäre, als in Wirklichkeit genutzt wird. Dies kann der Fall sein, wenn die Registriervariable DB2_SMS_TRUNC_TMPTABLE_THRESH verwendet wird.

Mit Hilfe des Datenbanksystemmonitors und der APIs zum Abfragen des Tabellenbereichs können Sie verfolgen, wie groß der verwendete Arbeitsbereich während des normalen Betriebsablaufs ist.

Zugehörige Konzepte:

- „Speicherbedarf für Datenbankobjekte" auf Seite 81

Zugehörige Referenzen:

- „sqlbmtsq - Table Space Query" in *Administrative API Reference*

Datenbankpartitionsgruppen

Eine *Datenbankpartitionsgruppe* ist eine Gruppe aus einer oder mehreren Datenbankpartitionen. Wenn Sie Tabellen für die Datenbank erstellen wollen, erstellen Sie zunächst die Datenbankpartitionsgruppe, in der die Tabellenbereiche gespeichert werden. Anschließend erstellen Sie den Tabellenbereich, in dem die Tabellen gespeichert werden.

Sie können benannte Untergruppen einer oder mehrerer Datenbankpartitionen in einer Datenbank definieren. Jede Untergruppe, die Sie definieren, wird als *Datenbankpartitionsgruppe* bezeichnet. Jede Untergruppe, die mehr als eine Datenbankpartition enthält, wird als *Datenbankpartitionsgruppe mit mehreren Partitionen* bezeichnet. Datenbankpartitionsgruppen mit mehreren Partitionen können nur mit Datenbankpartitionen definiert werden, die zum selben Exemplar gehören.

Abb. 25 auf Seite 92 zeigt ein Beispiel einer Datenbank mit fünf Partitionen mit folgenden Partitionsgruppen:

- Eine Datenbankpartitionsgruppe umfasst alle Datenbankpartitionen außer einer (Datenbankpartitionsgruppe 1).
- Eine Datenbankpartitionsgruppe enthält eine Datenbankpartition (Datenbankpartitionsgruppe 2).
- Eine Datenbankpartitionsgruppe enthält zwei Datenbankpartitionen (Datenbankpartitionsgruppe 3).
- Die Datenbankpartition innerhalb der Datenbankpartitionsgruppe 2 wird gemeinsam mit der Datenbankpartitionsgruppe 1 (und überlappend) benutzt.
- Es gibt eine einzelne Datenbankpartition innerhalb der Datenbankpartitionsgruppe 3, die gemeinsam mit Datenbankpartitionsgruppe 1 (und überlappend) benutzt wird.

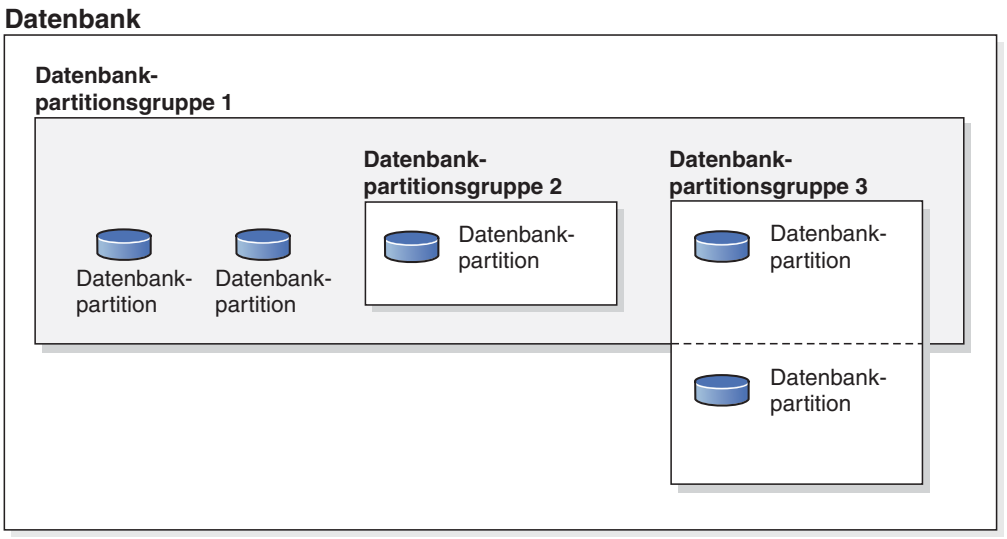


Abbildung 25. Datenbankpartitionsgruppen in einer Datenbank

Sie können eine neue Datenbankpartitionsgruppe mit Hilfe der Anweisung CREATE DATABASE PARTITION GROUP erstellen. Mit Hilfe der Anweisung ALTER DATABASE PARTITION GROUP lässt sie sich ändern. Die Daten werden auf alle Partitionen in einer Datenbankpartitionsgruppe verteilt, und Sie können eine oder mehrere Datenbankpartitionen einer Datenbankpartitionsgruppe hinzufügen oder aus ihr löschen. Wenn Sie eine Datenbankpartitionsgruppe mit mehreren Partitionen verwenden, müssen Sie eine Reihe von Gesichtspunkten im Hinblick auf den Aufbau der Datenbankpartitionsgruppe beachten.

Jede Datenbankpartition, die Teil einer Datenbanksystemkonfiguration ist, muss bereits in einer *Partitionskonfigurationsdatei* namens `db2nodes.cfg` definiert sein. Eine Datenbankpartitionsgruppe kann eine Datenbankpartition, mehrere Datenbankpartitionen und auch alle für das Datenbanksystem definierten Datenbankpartitionen enthalten.

Wenn eine Datenbankpartitionsgruppe erstellt oder geändert wird, wird ihr eine *Partitionierungszuordnung* zugeordnet. Die Partitionierungszuordnung wird vom Datenbankmanager in Verbindung mit einem *Partitionierungsschlüssel* und einem Hashalgorithmus verwendet, um festzulegen, in welcher Datenbankpartition der Datenbankpartitionsgruppe eine bestimmte Datenzeile gespeichert wird.

In einer nicht partitionierten Datenbank ist kein Partitionierungsschlüssel und keine Partitionierungszuordnung erforderlich. Eine Datenbankpartition ist ein Teil einer Datenbank, mit eigenen Benutzerdaten, Indizes, Konfigurationsdateien und Transaktionsprotokollen. Vom Datenbankmanager werden Standarddatenbankpartitionsgruppen verwendet, die erstellt werden, wenn die Datenbank erstellt wird. IBMCATGROUP ist die Standarddatenbankpartitionsgruppe für den Tabellenbereich, der die Systemkatalogtabellen enthält. IBMTEMPGROUP ist die Standarddatenbankpartitionsgruppe der Tabellenbereiche für temporäre Systemtabellen. IBMDEFAULTGROUP ist die Standarddatenbankpartitionsgruppe für die Tabellenbereiche, die die benutzerdefinierten Tabellen enthalten, die Sie dort speichern. Ein Tabellenbereich für temporäre Benutzertabellen für eine deklarierte temporäre Benutzertabelle kann in IBMDEFAULTGROUP bzw. in einer beliebigen benutzererstellten Datenbankpartitionsgruppe, jedoch nicht in IBMTEMPGROUP erstellt werden.

Zugehörige Konzepte:

- „Aufbau von Datenbankpartitionsgruppen" auf Seite 93
- „Partitionierungszuordnungen" auf Seite 94
- „Partitionierungsschlüssel" auf Seite 96

Zugehörige Referenzen:

- „ALTER DATABASE PARTITION GROUP statement" in *SQL Reference, Volume 2*
- „CREATE DATABASE PARTITION GROUP statement" in *SQL Reference, Volume 2*

Aufbau von Datenbankpartitionsgruppen

Wenn Sie eine nicht partitionierte Datenbank verwenden, spielen Überlegungen zum Aufbau von Datenbankpartitionsgruppen keine Rolle.

Der DB2®-Designadvisor ist ein Tool, dass zur Erstellung von Empfehlungen zu Datenbankpartitionsgruppen genutzt werden kann. Der Zugriff auf den DB2-Designadvisor ist über die Steuerzentrale sowie mit Hilfe des Befehls db2advis über den Befehlszeilenprozessor möglich.

Wenn Sie eine Datenbankpartitionsgruppe mit mehreren Partitionen verwenden, beachten Sie die folgenden Gesichtspunkte:

- In einer Datenbankpartitionsgruppe mit mehreren Partitionen können Sie nur dann einen eindeutigen Index erstellen, wenn er eine Obermenge des Partitionierungsschlüssels ist.
- Abhängig von der Anzahl der Datenbankpartitionen in der Datenbank können eine oder mehrere Datenbankpartitionsgruppen mit einer Partition und eine oder mehrere Datenbankpartitionsgruppen mit mehreren Partitionen vorhanden sein.
- Jeder Datenbankpartition muss eine eindeutige Partitionsnummer zugeordnet sein. Dieselbe Datenbankpartition kann in einer oder mehreren Datenbankpartitionsgruppen enthalten sein.
- Zur Gewährleistung einer schnellen Wiederherstellung der Datenbankpartition mit den Systemkatalogtabellen sollten Sie keine Benutzertabellen in derselben Datenbankpartition anlegen. Dies wird erreicht, wenn die Benutzertabellen in Datenbankpartitionsgruppen untergebracht werden, die nicht die Datenbankpartition der Datenbankpartitionsgruppe IBMCATGROUP enthalten.

Kleine Tabellen sollten in Datenbankpartitionsgruppen mit nur einer Partition angelegt werde, außer wenn Sie die Vorteile der *Kollokation* mit einer größeren Tabelle nutzen wollen. Unter Kollokation (Zusammenfassung) ist die Platzierung von Zeilen aus verschiedenen Tabellen, die zusammengehörige Daten enthalten, in die gleiche Datenbankpartition zu verstehen. Durch Kollokation zusammengefasste Tabellen ermöglichen DB2 Universal Database™ (DB2 UDB) den Einsatz effizienterer Verknüpfungsstrategien. Durch Kollokation zusammengefasste Tabellen können sich in einer Datenbankpartitionsgruppe mit einer Einzelpartition befinden. Tabellen werden als durch Kollokation zusammengefasst betrachtet, wenn sie sich in einer Datenbankpartitionsgruppe mit mehreren Partitionen befinden, dieselbe Anzahl von Spalten im Partitionierungsschlüssel haben und die Datentypen der sich entsprechenden Spalten partitionskompatibel sind. Zeilen in zusammengefassten Tabellen mit dem gleichen Wert im Partitionierungsschlüssel werden in derselben Datenbankpartition gespeichert. Tabellen können sich in separaten Tabellenbereichen in derselben Datenbankpartitionsgruppe befinden und trotzdem als durch Kollokation zusammengefasst betrachtet werden.

Mittelgroße Tabellen sollten nicht über zu viele Datenbankpartitionen verteilt werden. Zum Beispiel ist es möglich, dass eine 100-MB-Tabelle in einer Datenbankpartitionsgruppe mit 16 Partitionen zu besseren Leistungen führt als in einer Datenbankpartitionsgruppe mit 32 Partitionen.

Sie können Datenbankpartitionsgruppen dazu verwenden, OLTP-Tabellen (OLTP - Online-Transaktionsverarbeitung) von Entscheidungshilfetabellen (DSS-Tabellen - Decision Support System) zu trennen, um sicherzustellen, dass die Leistung von OLTP-Transaktionen nicht beeinträchtigt wird.

Zugehörige Konzepte:

- „Datenbankpartitionsgruppen" auf Seite 91
- „Partitionierungszuordnungen" auf Seite 94
- „Partitionierungsschlüssel" auf Seite 96
- „Tabellenkollokation" auf Seite 98
- „Partitionskompatibilität" auf Seite 98
- „Replizierte gespeicherte Abfragetabellen" auf Seite 99

Zugehörige Referenzen:

- „db2advis - DB2 Design Advisor Command" in *Command Reference*

Partitionierungszuordnungen

In einer Umgebung mit einer partitionierten Datenbank muss der Datenbankmanager eine Möglichkeit besitzen, zu wissen, welche Tabellenzeilen in welcher Datenbankpartition gespeichert werden. Der Datenbankmanager muss wissen, wo er die benötigten Daten findet. Zum Auffinden der Daten verwendet er ein Zuordnungsverzeichnis, das als *Partitionierungszuordnung* bezeichnet wird.

Eine Partitionierungszuordnung ist eine intern generierte Tabelle von entweder 4 096 Einträgen für Datenbankpartitionsgruppen mit mehreren Partitionen oder einem einzigen Eintrag für Datenbankpartitionsgruppen mit einer Einzelpartition. Für eine Datenbankpartitionsgruppe mit nur einer Partition enthält die Partitionierungszuordnung nur einen Eintrag mit der Partitionsnummer der Datenbankpartition, in der alle Zeilen einer Datenbanktabelle gespeichert werden. Für Datenbankpartitionsgruppen mit mehreren Partitionen werden die Partitionsnummern der Datenbankpartitionsgruppe immer wiederholt reihum angegeben. Vergleichbar mit der Verwendung einer Stadtkarte, die durch Gitterlinien in Sektoren unterteilt ist, verwendet der Datenbankmanager einen *Partitionierungsschlüssel*, um die Speicherposition (die Datenbankpartition) zu bestimmen, in der die Daten gespeichert werden.

Nehmen Sie zum Beispiel an, dass Sie eine Datenbank in vier Datenbankpartitionen (mit den Nummern 0–3) erstellt haben. Die Partitionierungszuordnung für die Datenbankpartitionsgruppe IBMDEFAULTGROUP dieser Datenbank wäre wie folgt:

```
0 1 2 3 0 1 2 ...
```

Wenn eine Datenbankpartitionsgruppe in der Datenbank mit den Datenbankpartitionen 1 und 2 erstellt würde, sähe die Partitionierungszuordnung für diese Datenbankpartitionsgruppe wie folgt aus:

```
1 2 1 2 1 2 1 ...
```

Wenn der Partitionierungsschlüssel für eine in die Datenbank zu ladende Tabelle eine ganze Zahl (Integer) mit möglichen Werten zwischen 1 und 500 000 ist, wird der Partitionierungsschlüssel mit einem Hashverfahren auf eine Partitionsnummer zwischen 0 und 4 095 abgebildet. Diese Nummer wird als Index für die Partitionierungszuordnung verwendet, um die Datenbankpartition für die betreffende Zeile auszuwählen.

Abb. 26 zeigt, wie der Zeile mit dem Partitionierungsschlüsselwert (c1, c2, c3) die Partitionsnummer 2 zugeordnet wird, die ihrerseits auf Datenbankpartition n5 verweist.

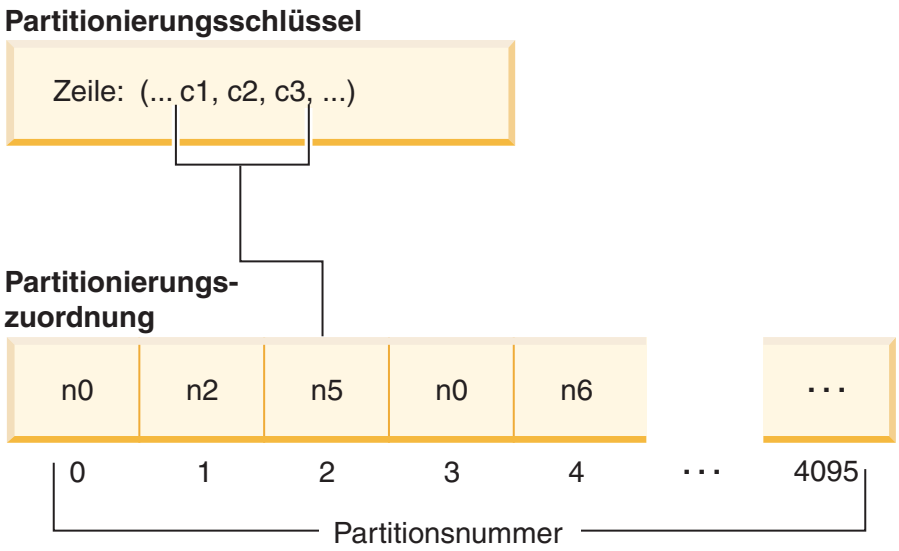


Abbildung 26. Datenverteilung mit einer Partitionierungszuordnung

Eine solche Partitionierungszuordnung ist eine flexible Steuermethode für die Speicherung von Daten in einer partitionierten Datenbank. Wenn zu einem zukünftigen Zeitpunkt die Notwendigkeit eintritt, die Datenverteilung auf die Datenbankpartitionen Ihrer Datenbank zu ändern, können Sie dazu das Dienstprogramm zur Datenumverteilung verwenden. Dieses Dienstprogramm ermöglicht Ihnen, die Datenverteilung neu auszugleichen oder bewusst ungleichmäßig zu gestalten.

Mit Hilfe der API **sqlugtpi** zum Abrufen von Informationen zur Tabellenpartitionierung können Sie eine Kopie der Partitionierungszuordnung abrufen, die Sie anzeigen können.

Zugehörige Konzepte:

- „Datenbankpartitionsgruppen" auf Seite 91
- „Aufbau von Datenbankpartitionsgruppen" auf Seite 93
- „Partitionierungsschlüssel" auf Seite 96

Zugehörige Referenzen:

- „sqlugtpi - Get Table Partitioning Information" in *Administrative API Reference*

Partitionierungsschlüssel

Ein *Partitionierungsschlüssel* ist eine Spalte (oder Spaltengruppe), die dazu verwendet wird, die Partition zu bestimmen, in der eine bestimmte Datenzeile gespeichert wird. Ein Partitionierungsschlüssel wird in einer Tabelle mit Hilfe der Anweisung CREATE TABLE definiert. Wenn kein Partitionierungsschlüssel für eine Tabelle in einem Tabellenbereich, der über mehr als eine Datenbankpartition in einer Datenbankpartitionsgruppe verteilt ist, definiert wird, wird ein Partitionierungsschlüssel standardmäßig aus der ersten Spalte des Primärschlüssels erstellt. Wenn kein Primärschlüssel angegeben ist, wird standardmäßig die erste Spalte der Tabelle, die keine langen Daten enthält, als Partitionierungsschlüssel angenommen. (*Lang* beinhaltet hier alle langen Datentypen (LONG) und alle LOB-Datentypen). Wenn Sie eine Tabelle in einem Tabellenbereich erstellen, der zu einer Datenbankpartitionsgruppe mit nur einer Partition gehört, und einen Partitionierungsschlüssel erstellen wollen, müssen Sie den Partitionierungsschlüssel explizit definieren. In diesem Fall wird kein Standardschlüssel erstellt.

Wenn keine Spalte die Voraussetzungen für einen standardmäßig erstellten Partitionierungsschlüssel erfüllt, wird die Tabelle ohne Partitionierungsschlüssel erstellt. Tabellen ohne Partitionierungsschlüssel sind nur in Datenbankpartitionsgruppen mit einer Partition zulässig. Sie können Partitionierungsschlüssel auch später noch mit der Anweisung ALTER TABLE hinzufügen oder entfernen. Das Ändern des Partitionierungsschlüssels ist nur möglich in einer Tabelle, deren Tabellenbereich zu einer Datenbankpartitionsgruppe mit einer Einzelpartition gehört.

Die Auswahl eines guten Partitionierungsschlüssels ist von großer Bedeutung. Sie sollten sich über Folgendes im Klaren sein:

- Wie der Zugriff auf Tabellen erfolgen soll
- Die Art der Abfrageauslastung
- Die vom Datenbanksystem angewendeten Verknüpfungsstrategien

Wenn Kollokation kein Hauptgesichtspunkt ist, dann zeichnet sich ein guter Partitionierungsschlüssel für eine Tabelle dadurch aus, dass er die Daten gleichmäßig über alle Datenbankpartitionen in der Datenbankpartitionsgruppe verteilt. Der Partitionierungsschlüssel jeder Tabelle in einem Tabellenbereich, der einer Datenbankpartitionsgruppe zugeordnet ist, bestimmt, ob die Tabellen durch Kollokation zusammengefasst werden. Tabellen werden als zusammengefasst betrachtet, wenn folgende Bedingungen gelten:

- Die Tabellen liegen in Tabellenbereichen, die in derselben Datenbankpartitionsgruppe sind.
- Die Partitionierungsschlüssel in jeder Tabelle haben dieselbe Anzahl von Spalten.
- Die Datentypen der entsprechenden Spalten sind partitionskompatibel.

Diese Merkmale stellen sicher, dass Zeilen zusammengefasster Tabellen mit denselben Werten für ihre Partitionierungsschlüssel in derselben Partition gespeichert werden.

Ein ungeeigneter Partitionierungsschlüssel kann zu einer ungleichmäßigen Datenverteilung führen. Spalten mit ungleichmäßig verteilten Daten und Spalten mit einer kleinen Anzahl unterschiedlicher Werte sollten nicht als Partitionierungsschlüssel ausgewählt werden. Die Anzahl der unterschiedlichen Werte muss ausreichend groß sein, um eine gleichmäßige Verteilung der Zeilen auf alle Datenbankpartitionen in der Datenbankpartitionsgruppe sicherzustellen. Der Aufwand für die Anwendung des Hashalgorithmus zur Partitionierung ist proportional zur Größe

des Partitionierungsschlüssels. Der Partitionierungsschlüssel kann nicht mehr als 16 Spalten enthalten. Jedoch wird bei weniger Spalten eine bessere Leistung erzielt. Nicht benötigte Spalten sollten daher nicht in den Partitionierungsschlüssel aufgenommen werden.

Die folgenden Punkte sollten bei der Definition von Partitionierungsschlüsseln beachtet werden:

- Die Erstellung einer Mehrpartitionstabelle, die ausschließlich lange Datentypen (LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB oder DBCLOB) enthält, wird wird nicht unterstützt.
- Die Definition des Partitionierungsschlüssels kann nicht geändert werden.
- Der Partitionierungsschlüssel sollte die am häufigsten an Verknüpfungen beteiligten Spalten enthalten.
- Der Partitionierungsschlüssel sollte außerdem aus Spalten bestehen, die häufig von Klauseln GROUP BY betroffen sind.
- Jeder eindeutige Schlüssel oder Primärschlüssel muss alle Spalten des Partitionierungsschlüssels enthalten.
- In einer OLTP-Umgebung (Online-Transaktionsverarbeitung) sollten alle Spalten des Partitionierungsschlüssels an der Transaktion über Gleichheitsvergleichselemente (=) mit Konstanten oder Hostvariablen beteiligt sein. Nehmen Sie zum Beispiel an, Sie haben eine Personalnummer *emp_no*, die in Transaktionen häufig wie folgt oder ähnlich verwendet wird:

  ```
  UPDATE emp_table SET ... WHERE
  emp_no = hostvariable
  ```

  In diesem Fall wäre die Spalte EMP_NO ein guter einspaltiger Partitionierungsschlüssel für die Tabelle EMP_TABLE.

Hashpartitionierung heißt die Methode, mit der die Speicherposition jeder Zeile in der partitionierten Tabelle ermittelt wird. Diese Methode funktioniert folgendermaßen:

1. Der Hashalgorithmus wird auf den Wert des Partitionierungsschlüssels angewendet und generiert eine Partitionsnummer zwischen 0 und 4095.
2. Die Partitionierungszuordnung wird bei der Erstellung einer Datenbankpartitionsgruppe erstellt. Die Partitionsnummern werden immer wieder in derselben Reihenfolge nacheinander wiederholt, bis die Partitionierungszuordnung gefüllt ist.
3. Die Partitionsnummer wird als Index für die Position in der Partitionierungszuordnung verwendet. Die Nummer an dieser Position in der Partitionierungszuordnung ist die Nummer der Datenbankpartition, in der die Zeile gespeichert wird.

Zugehörige Konzepte:

- „Datenbankpartitionsgruppen" auf Seite 91
- „Aufbau von Datenbankpartitionsgruppen" auf Seite 93
- „Partitionierungszuordnungen" auf Seite 94
- „Der Designadvisor" in *Systemverwaltung: Optimierung*

Zugehörige Referenzen:

- „ALTER TABLE statement" in *SQL Reference, Volume 2*

Tabellenkollokation

Sie stellen möglicherweise fest, dass zwei oder mehr Tabellen häufig zusammen Daten als Antwort für bestimmte Abfragen liefern. In diesem Fall ist es sinnvoll, zusammengehörige Daten aus solchen Tabellen möglichst nah beieinander zu speichern. In einer Umgebung, in der die Datenbank physisch auf zwei oder mehr Datenbankpartitionen verteilt ist, muss es eine Möglichkeit geben, zusammengehörige Teile der verteilten Tabellen so nah wie möglich beieinander zu halten. Diese Möglichkeit wird als *Tabellenkollokation* bezeichnet.

Tabellen sind in einer Kollokation zusammengefasst, wenn sie in derselben Datenbankpartitionsgruppe gespeichert sind und ihre Partitionierungsschlüssel kompatibel sind. Durch Speichern beider Tabellen in derselben Datenbankpartitionsgruppe wird sichergestellt, dass sie eine gemeinsame Partitionierungszuordnung haben. Die Tabellen können sich in verschiedenen Tabellenbereichen befinden, jedoch müssen die Tabellenbereiche derselben Datenbankpartitionsgruppe zugeordnet sein. Die Datentypen der entsprechenden Spalten in den jeweiligen Partitionierungsschlüsseln müssen *partitionskompatibel* sein.

DB2® Universal Database (DB2 UDB) kann beim Zugriff auf mehr als eine Tabelle bei einer Verknüpfung oder einer Unterabfrage erkennen, dass sich die zu verknüpfenden Daten in derselben Datenbankpartition befinden. Wenn dies geschieht, kann DB2 entscheiden, die Verknüpfung oder Unterabfrage in der Datenbankpartition auszuführen, in der die Daten gespeichert sind, anstatt die Daten zwischen Datenbankpartitionen auszutauschen. Diese Möglichkeit, Verknüpfungen oder Unterabfragen in der Datenbankpartition auszuführen, bietet wesentliche Leistungsvorteile.

Zugehörige Konzepte:

- „Datenbankpartitionsgruppen" auf Seite 91
- „Aufbau von Datenbankpartitionsgruppen" auf Seite 93
- „Partitionierungsschlüssel" auf Seite 96
- „Partitionskompatibilität" auf Seite 98

Partitionskompatibilität

Die Basisdatentypen entsprechender Spalten von Partitionierungsschlüsseln werden verglichen und können als *partitionskompatibel* deklariert werden. Partitionskompatible Datentypen haben die Eigenschaft, dass ein bestimmter Partitionierungsalgorithmus zwei Variablen mit jeweils einem dieser Datentypen dieselbe Partitionsnummer zuordnet, wenn sie denselben Wert haben.

Partitionskompatibilität hat folgende Merkmale:

- Ein Wert des Basisdatentyps ist mit einem anderen desselben Basistyps kompatibel.
- Interne Formate werden für die Datentypen DATE (Datum), TIME (Uhrzeit) und TIMESTAMP (Zeitmarke) verwendet. Sie sind untereinander nicht kompatibel, und keiner dieser Typen ist mit dem Zeichendatentyp (CHAR) kompatibel.
- Die Partitionskompatibilität wird von Spalten mit den Definitionen NOT NULL oder FOR BIT DATA nicht berührt.
- Nullwerte (NULL) kompatibler Datentypen werden auf identische Weise behandelt, Nullwerte nicht kompatibler Datentypen möglicherweise nicht.

- Die Basisdatentypen eines benutzerdefinierten Datentyps werden zur Analyse der Partitionskompatibilität verwendet.
- Dezimalzahlen desselben Werts im Partitionierungsschlüssel werden identisch behandelt, auch wenn ihre Anzahl an Kommastellen und ihre Genauigkeit unterschiedlich sind.
- Folgende Leerzeichen in Zeichenfolgen (CHAR, VARCHAR, GRAPHIC oder VARGRAPHIC) werden vom Hashalgorithmus ignoriert.
- BIGINT, SMALLINT und INTEGER sind kompatible Datentypen.
- REAL und FLOAT sind kompatible Datentypen.
- CHAR und VARCHAR verschiedener Längen sind kompatible Datentypen.
- GRAPHIC und VARGRAPHIC sind kompatible Datentypen.
- Die Partitionskompatibilität gilt nicht für die Datentypen LONG VARCHAR, LONG VARGRAPHIC, CLOB, DBCLOB und BLOB, da sie in Partitionierungsschlüsseln nicht unterstützt werden.

Zugehörige Konzepte:

- „Datenbankpartitionsgruppen" auf Seite 91
- „Aufbau von Datenbankpartitionsgruppen" auf Seite 93
- „Partitionierungsschlüssel" auf Seite 96

Replizierte gespeicherte Abfragetabellen

Eine *gespeicherte Abfragetabelle* ist eine Tabelle, die durch eine Abfrage definiert ist, mit der auch die Daten für die Tabelle festgelegt werden. Durch gespeicherte Abfragetabellen kann die Leistungsfähigkeit von Abfragen erhöht werden. Wenn DB2® Universal Database (DB2 UDB) erkennt, dass ein Teil einer Abfrage durch eine gespeicherte Abfragetabelle aufgelöst werden könnte, kann der Datenbankmanager die Abfrage so abwandeln, dass die entsprechende gespeicherte Abfragetabelle verwendet wird.

In einer Umgebung mit partitionierten Datenbanken können Sie gespeicherte Abfragetabellen replizieren. Durch die Verwendung *replizierter gespeicherter Abfragetabellen* können Sie die Leistung von Abfragen erhöhen. Eine replizierte gespeicherte Abfragetabelle basiert auf einer Tabelle, die möglicherweise in einer Datenbankpartitionsgruppe mit einer Einzelpartition erstellt wurde, die Sie jedoch über alle Datenbankpartitionen in einer anderen Datenbankpartitionsgruppe replizieren wollen. Die replizierte gespeicherte Abfragetabelle wird durch Ausführen der Anweisung CREATE TABLE mit dem Schlüsselwort REPLICATED erstellt.

Mit Hilfe von replizierten gespeicherten Abfragetabellen können Sie Kollokationen zwischen Tabellen herstellen, die normalerweise nicht kollokiert sind. Replizierte gespeicherte Abfragetabellen sind besonders nützlich für Verknüpfungen großer Fakttabellen mit kleinen Dimensionstabellen. Zur Minimierung des erforderlichen zusätzlichen Speicherbedarfs sowie des Aufwands zum Aktualisieren aller Replikate, sollten zu replizierende Tabellen klein sein und nicht häufig aktualisiert werden.

Anmerkung: Sie sollten auch in Erwägung ziehen, umfangreiche Tabellen, die selten aktualisiert werden, zu replizieren: Der einmalige Zusatzaufwand für eine Replikation wird durch die mit Hilfe der Kollokation erzielten Leistungsvorteile wettgemacht.

Durch Angeben eines geeigneten Vergleichselements in der Unterauswahlklausel, die zum Definieren der replizierten Tabelle verwendet wird, können Sie ausgewählte Spalten, ausgewählte Zeilen (oder beides) replizieren.

Zugehörige Konzepte:

- „Aufbau von Datenbankpartitionsgruppen" auf Seite 93
- „Der Designadvisor" in *Systemverwaltung: Optimierung*

Zugehörige Tasks:

- „Erstellen einer gespeicherten Abfragetabelle" in *Systemverwaltung: Implementierung*

Zugehörige Referenzen:

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

Aufbau von Tabellenbereichen

Ein Tabellenbereich ist eine Speicherstruktur, die Tabellen, Indizes, große Objekte (LOB-Daten) und Langfelddaten (LONG-Daten) enthält. Tabellenbereiche befinden sich in Datenbankpartitionsgruppen. Mit ihrer Hilfe können Sie die Speicherposition der Datenbank- und Tabellendaten direkt bestimmten Behältern zuweisen. (Ein Behälter kann ein Verzeichnisname, Einheitenname oder Dateiname sein.) Die möglichen Vorzüge zeigen sich in einer besseren Leistung und einer flexibleren Konfiguration.

Da sich Tabellenbereiche in Datenbankpartitionsgruppen befinden, bestimmt der zur Speicherung einer Tabelle ausgewählte Tabellenbereich, wie die Daten dieser Tabelle auf die Datenbankpartitionen in einer Datenbankpartitionsgruppe verteilt werden. Ein einzelner Tabellenbereich kann sich über mehrere Behälter erstrecken. Mehrere Behälter (eines oder mehrerer Tabellenbereiche) können auf derselben physischen Platte (bzw. auf demselben Laufwerk) erstellt werden. Zur Erzielung einer optimalen Leistung sollte jeder Behälter eine andere Platte verwenden. Abb. 27 auf Seite 101 zeigt ein Beispiel für die Anordnungsbeziehung zwischen Tabellen und Tabellenbereichen innerhalb einer Datenbank und den Behältern, die dieser Datenbank zugeordnet sind.

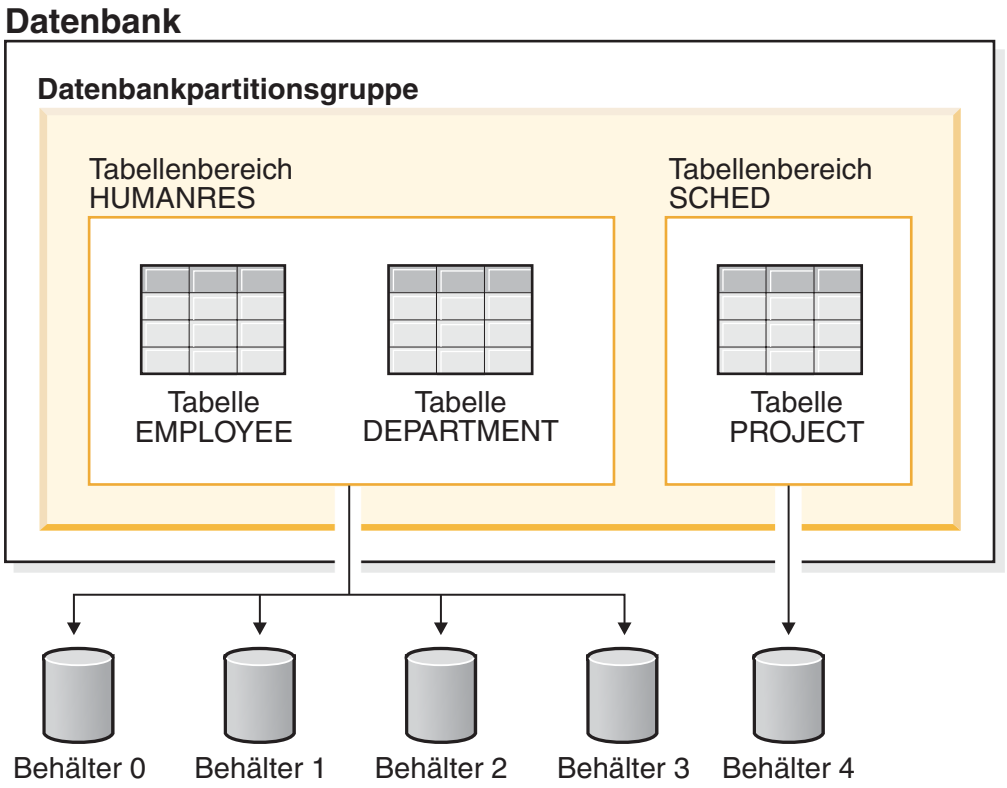


Abbildung 27. Tabellenbereiche und Tabellen in einer Datenbank

Die Tabellen EMPLOYEE und DEPARTMENT befinden Sie im Tabellenbereich HUMANRES, der sich über die Behälter 0, 1, 2 und 3 erstreckt. Die Tabelle PROJECT befindet sich im Tabellenbereich SCHED im Behälter 4. Dieses Beispiel zeigt jeden Behälter auf einer getrennten Platte.

Der Datenbankmanager versucht, die Datenmenge möglichst gleichmäßig über die Behälter zu verteilen. Das heißt, es werden alle Behälter zum Speichern der Daten verwendet. Die Anzahl der Seiten, die der Datenbankmanager in einen Behälter schreibt, bevor er einen anderen Behälter verwendet, wird mit dem Parameter *EXTENTSIZE* definiert. Der Datenbankmanager beginnt beim Speichern der Tabellendaten nicht immer mit dem ersten Behälter.

Abb. 28 auf Seite 102 zeigt den Tabellenbereich HUMANRES mit einem Wert von zwei 4-KB-Seiten für EXTENTSIZE und vier Behältern mit einer kleinen Anzahl zugeordneter Speicherbereiche. Die Tabellen DEPARTMENT und EMPLOYEE haben jeweils sieben Seiten und verteilen sich über alle vier Behälter.

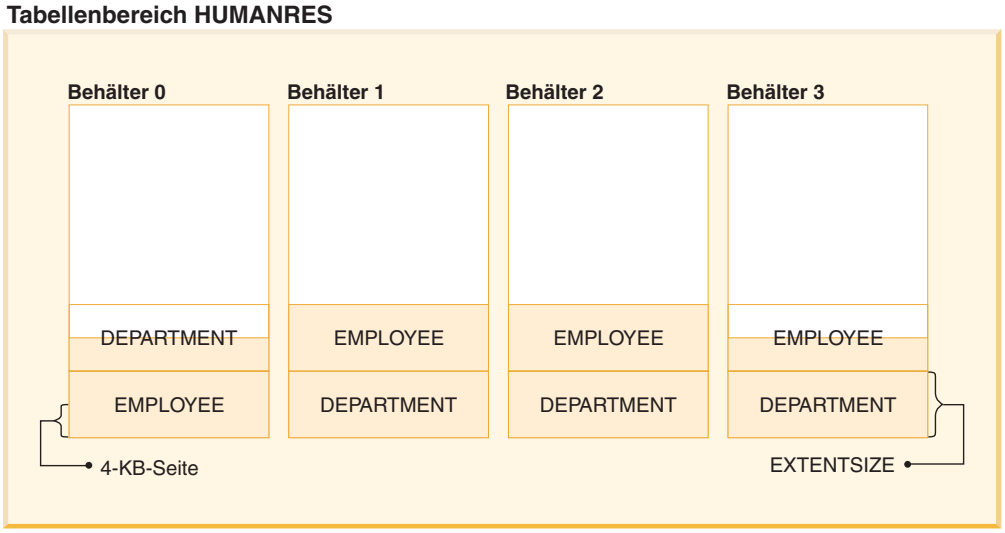


Abbildung 28. Behälter und EXTENTSIZE große Speicherbereiche in einem Tabellenbereich

Eine Datenbank muss mindestens drei Tabellenbereiche enthalten:

- Einen *Katalogtabellenbereich*, der alle Systemkatalogtabellen für die Datenbank enthält. Dieser Tabellenbereich heißt SYSCATSPACE und kann nicht gelöscht werden. Die Standarddatenbankpartitionsgruppe für diesen Tabellenbereich heißt IBMCATGROUP.
- Einen oder mehrere *Benutzertabellenbereiche*, die alle benutzerdefinierten Tabellen enthalten. Standardmäßig wird ein Tabellenbereich namens USERSPACE1 erstellt. Die Standarddatenbankpartitionsgruppe für diesen Tabellenbereich heißt IBMDEFAULTGROUP.

  Beim Erstellen einer Tabelle sollten Sie einen Tabellenbereichsnamen angeben, andernfalls kann es zu unerwünschten Ergebnissen kommen.

  Die Seitengröße für eine Tabelle wird entweder durch die Zeilengröße oder durch die Anzahl von Spalten bestimmt. Die für eine Zeile maximal zulässige Länge hängt von der Seitengröße des Tabellenbereichs ab, in dem die Tabelle erstellt wird. Gültige Werte für die Seitengröße sind 4 KB (Standardwert), 8 KB, 16 KB und 32 KB. Sie können einen Tabellenbereich mit einer Seitengröße für die Basistabelle und einen weiteren Tabellenbereich mit einer anderen Seitengröße für lange Daten (LONG) oder LOB-Daten verwenden. (Hier ist wiederum zu beachten, dass SMS keine Tabellen unterstützt, die sich über mehrere Tabellenbereiche erstrecken, im Gegensatz zu DMS.) Wenn die Anzahl der Spalten oder die Zeilengröße die Grenze für die Seitengröße eines Tabellenbereichs überschreitet, wird ein Fehler zurückgegeben (SQLSTATE 42997).
- Einen oder mehrere *temporäre Tabellenbereiche*, die temporäre Tabellen enthalten. Temporäre Tabellenbereiche können *temporäre Systemtabellenbereiche* oder *temporäre Benutzertabellenbereiche* sein. Eine Datenbank muss mindestens über einen temporären Tabellenbereich verfügen. Standardmäßig wird ein temporärer Systemtabellenbereich namens TEMPSPACE1 bei der Erstellung der Datenbank erstellt. Die Standarddatenbankpartitionsgruppe für diesen Tabellenbereich heißt IBMTEMPGROUP. Temporäre Benutzertabellenbereiche werden *nicht* standardmäßig bei der Erstellung einer Datenbank erstellt.

  Wenn eine Datenbank mehr als einen temporären Tabellenbereich verwendet und ein neues temporäres Objekt benötigt wird, wählt das Optimierungsprogramm eine geeignete Seitengröße für dieses Objekt aus. Anschließend wird dieses Objekt dem temporären Tabellenbereich mit der entsprechenden Seitengröße zugeordnet. Wenn mehr als ein temporärer Tabellenbereich mit dieser

Seitengröße vorhanden ist, werden die Tabellenbereiche reihum ausgewählt. In den meisten Fällen ist es nicht empfehlenswert, mehr als einen temporären Tabellenbereich für eine bestimmte Seitengröße zu haben.

Wenn Abfragen auf Tabellen in Tabellenbereichen ausgeführt werden, die mit einer größeren Seitengröße als dem Standardwert von 4 KB definiert sind (z. B. eine Klausel ORDER BY auf 1012 Spalten), schlagen manche dieser Abfragen möglicherweise fehl. Dies geschieht, wenn keine temporären Tabellenbereiche mit einer größeren Seitengröße definiert sind. Möglicherweise müssen Sie einen temporären Tabellenbereich mit einer größeren Seitengröße (8 KB, 16 KB oder 32 KB) erstellen. Jede DML-Anweisung (Datenbearbeitungssprache) könnte fehlschlagen, sofern kein temporärer Tabellenbereich mit derselben Seitengröße wie die größte in dem Benutzertabellenbereich verwendete Seitengröße vorhanden ist.

Sie sollten einen einzelnen temporären SMS-Tabellenbereich definieren, dessen Seitengröße der Seitengröße entspricht, die in den meisten der Benutzertabellenbereiche verwendet wird. Dies sollte für typische Umgebungen und Auslastungen angemessen sein.

In einer Umgebung mit partitionierten Datenbanken verfügt der Katalogknoten über alle drei Tabellenbereiche, während die anderen Datenbankpartitionen nur die Tabellenbereiche TEMPSPACE1 und USERSPACE1 enthalten. Es gibt zwei Typen von Tabellenbereichen, die beide in einer einzelnen Datenbank verwendet werden können:

- Vom System verwaltete Bereiche (SMS - System Managed Space), bei denen der Dateimanager des Betriebssystems den Speicherbereich steuert
- Von der Datenbank verwaltete Bereiche (DMS - Database Managed Space), bei denen der Datenbankmanager den Speicherbereich steuert.

Zugehörige Konzepte:

- „Table spaces and other storage structures" in *SQL Reference, Volume 1*
- „Vom Betriebssystem verwalteter Speicherbereich (SMS)" auf Seite 104
- „Von der Datenbank verwalteter Speicherbereich (DMS)" auf Seite 106
- „SMS- und DMS-Tabellenbereiche im Vergleich" auf Seite 125
- „Platten-E/A für Tabellenbereiche" auf Seite 126
- „Überlegungen zur Auslastung beim Entwurf von Tabellenbereichen" auf Seite 128
- „Parameter EXTENTSIZE" auf Seite 130
- „Beziehung zwischen Tabellenbereichen und Pufferpools" auf Seite 131
- „Beziehung zwischen Tabellenbereichen und Datenbankpartitionsgruppen" auf Seite 132
- „Entwurf temporärer Tabellenbereiche" auf Seite 145
- „Entwurf von Katalogtabellenbereichen" auf Seite 148

Zugehörige Tasks:

- „Erstellen eines Tabellenbereichs" in *Systemverwaltung: Implementierung*
- „Optimieren der Leistung von Tabellenbereichen bei Datenspeicherung auf RAID-Einheiten" auf Seite 148

Zugehörige Referenzen:

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

Vom Betriebssystem verwalteter Speicherbereich (SMS)

In einem vom Betriebssystem verwalteten Tabellenbereich (SMS - System Managed Space) ordnet der Dateisystemmanager des Betriebssystems den Speicherbereich zu, in dem die Tabelle gespeichert wird, und verwaltet diesen Bereich. Das Speichermodell enthält in der Regel viele Dateien, die Tabellenobjekte darstellen, die im Speicherbereich des Dateisystems gespeichert sind. Der Benutzer entscheidet über die Speicherposition der Dateien, DB2® Universal Database (DB2 UDB) verwaltet ihre Namen, und das Dateisystem ist für ihre Verwaltung auf dem System zuständig. Durch Steuern der Datenmenge, die in jede Datei geschrieben wird, verteilt der Datenbankmanager die Daten gleichmäßig auf die Behälter der Tabellenbereiche. Standardmäßig sind die ersten Tabellenbereiche, die zum Zeitpunkt der Erstellung einer Datenbank erstellt werden, SMS-Tabellenbereiche.

Jeder Tabelle ist mindestens eine physische SMS-Datei zugeordnet.

Wenn Sie eine bessere Einfügeleistung benötigen, sollten Sie die Aktivierung der mehrseitigen Dateizuordnung in Betracht ziehen. Dadurch kann das System der Datei gleichzeitig einen ganzen EXTENTSIZE großen Bereich anstatt einer Seite zuordnen bzw. die Datei um einen solchen Bereich erweitern. Wenn Sie planen, mehrdimensionale Tabellen (MDC-Tabellen) in Ihrem SMS-Tabellenbereich zu speichern, sollten Sie die mehrseitige Dateizuordnung aktivieren. Ab Version 8.2 wird bei der Erstellung einer Datenbank (auch einer partitionierten Datenbank) die mehrseitige Dateizuordnung standardmäßig aktiviert. Jedoch ist die mehrseitige Dateizuordnung vielleicht nicht die Standardeinstellung bei der Erstellung einer neuen Datenbank, wenn Sie die Registriervariable DB2_NO_MPFA_FOR_NEW_DB aktiviert haben. Zur Aktivierung der mehrseitigen Dateizuordnung für eine Datenbank wird das Dienstprogramm `db2empfa` verwendet. In einer Umgebung mit partitionierten Datenbanken muss dieses Dienstprogramm in jeder Datenbankpartition ausgeführt werden. Wenn die mehrseitige Dateizuordnung aktiviert ist, kann sie nicht mehr inaktiviert werden.

SMS-Tabellenbereiche werden über die Option MANAGED BY SYSTEM im Befehl CREATE DATABASE oder in der Anweisung CREATE TABLESPACE definiert. Dabei müssen Sie zwei Schlüsselfaktoren beim Entwurf Ihrer SMS-Tabellenbereiche beachten:

- Behälter für den Tabellenbereich

  Sie müssen die Anzahl der Behälter angeben, die Sie für Ihren Tabellenbereich verwenden wollen. Es ist sehr wichtig, alle gewünschten Behälter zu ermitteln, weil Sie nach dem Erstellen des Tabellenbereichs keine Behälter löschen oder hinzufügen können. In einer partitionierten Datenbankumgebung kann die Anweisung ALTER TABLESPACE verwendet werden, um beim Hinzufügen einer neuen Partition zu einer Datenbankpartitionsgruppe eines SMS-Tabellenbereichs Behälter für die neue Partition hinzuzufügen.

  Jeder Behälter, der für einen SMS-Tabellenbereich verwendet wird, gibt einen absoluten oder relativen Verzeichnisnamen an. Jedes dieser Verzeichnisse kann sich auf einem anderen Dateisystem (oder einer anderen physischen Platte) befinden. Die Maximalgröße des Tabellenbereichs kann wie folgt abgeschätzt werden:

  ```
  Anzahl von Behältern * (maximale Dateisystemgröße, die vom
     Betriebssystem unterstützt wird)
  ```

Diese Formel setzt voraus, dass jedem Behälter ein bestimmtes Dateisystem zugeordnet wird und dass für jedes Dateisystem der maximale Speicherbereich verfügbar ist. In der Praxis ist dies möglicherweise nicht der Fall, und die Maximalgröße des Tabellenbereichs kann sehr viel kleiner sein. Darüber hinaus bestehen auch Einschränkungen durch SQL bezüglich der Größe von Datenbankobjekten, die sich auf die maximale Größe eines Tabellenbereichs auswirken können.

Anmerkung: Gehen Sie beim Definieren der Behälter mit besonderer Sorgfalt vor. Wenn die Behälter bereits Dateien oder Verzeichnisse enthalten, wird eine Fehlernachricht (SQL0298N) zurückgegeben.

- Parameter EXTENTSIZE für den Tabellenbereich

  Der Wert für den Parameter EXTENTSIZE kann nur beim Erstellen des Tabellenbereichs angegeben werden. Da spätere Änderungen daran nicht möglich sind, ist es wichtig, einen geeigneten Wert für EXTENTSIZE anzugeben.

  Wenn Sie beim Erstellen eines Tabellenbereichs für den Parameter EXTENTSIZE keinen Wert angeben, erstellt der Datenbankmanager den Tabellenbereich mit dem Standardwert, der durch den Konfigurationsparameter *dft_extent_sz* der Datenbank definiert ist. Dieser Konfigurationsparameter wird anfangs auf der Grundlage der Informationen gesetzt, die beim Erstellen der Datenbank angegeben werden. Wird der Parameter *dft_extent_sz* nicht mit dem Befehl CREATE DATABASE angegeben, wird der Standardwert auf 32 gesetzt.

Um geeignete Werte für die Anzahl der Behälter und für den Parameter EXTENTSIZE für den Tabellenbereich festlegen zu können, benötigen Sie folgende Kenntnisse:

- Den oberen Grenzwert, den Ihr Betriebssystem für die Größe eines logischen Dateisystems festlegt

  Zum Beispiel haben einige Betriebssysteme eine obere Begrenzung von 2 GB. Wenn Sie also ein Tabellenobjekt mit einer Größe von 64 GB erstellen möchten, benötigen Sie auf dieser Art System mindestens 32 Behälter.

  Wenn Sie einen Tabellenbereich erstellen, können Sie Behälter angeben, die sich auf verschiedenen Dateisystemen befinden, und dadurch die Menge der Daten erhöhen, die in der Datenbank gespeichert werden können.

- Die Art und Weise, wie der Datenbankmanager die einem Tabellenbereich zugeordneten Datendateien und Behälter verwaltet

  Die erste Datei mit Tabellendaten (`SQL00001.DAT`) wird im ersten für den Tabellenbereich angegebenen Behälter erstellt. Diese Datei darf so weit anwachsen, bis sie die durch den Wert für EXTENTSIZE festgelegte Größe erreicht. Nach Erreichen dieser Größe schreibt der Datenbankmanager Daten in die Datei `SQL00001.DAT` im nächsten Behälter. Dieser Prozess wird fortgesetzt, bis alle Behälter Dateien des Namens `SQL00001.DAT` enthalten. Wenn dies eintritt, kehrt der Datenbankmanager zum ersten Behälter zurück. Dieser Prozess (der auch als *Striping - einheitenübergreifendes Lesen und Schreiben von Daten* bezeichnet wird) wird über die Behälterverzeichnisse fortgesetzt, bis ein Behälter voll ist (SQL0289N) oder vom Betriebssystem kein weiterer Speicherbereich mehr zugeordnet werden kann (Fehlernachricht: Datenträger voll).Das Striping-Verfahren wird auch für Index- (`SQL`*`nnnnn`*`.INX`), Langfeld- (`SQL`*`nnnnn`*`.LF`) und LOB-Dateien (`SQL`*`nnnnn`*`.LB` und `SQL`*`nnnnn`*`.LBA`) verwendet.

  Anmerkung: Der SMS-Tabellenbereich ist voll, sobald irgendeiner seiner Behälter voll ist. Daher ist es wichtig, dass jedem Behälter dieselbe Menge an Speicherbereich zur Verfügung steht.

Um die Daten gleichmäßiger über die Behälter zu verteilen, bestimmt der Datenbankmanager den Behälter, der zuerst verwendet werden soll, indem er den Wert der Tabellenkennung (`SQL00001.DAT` im obigen Beispiel) und die Anzahl der Behälter als Faktoren verwendet. Die Behälter werden beginnend mit dem Wert 0 durchnummeriert.

Zugehörige Konzepte:

- „Aufbau von Tabellenbereichen" auf Seite 100
- „Von der Datenbank verwalteter Speicherbereich (DMS)" auf Seite 106
- „SMS- und DMS-Tabellenbereiche im Vergleich" auf Seite 125

Zugehörige Referenzen:

- „db2empfa - Enable Multipage File Allocation Command" in *Command Reference*

Von der Datenbank verwalteter Speicherbereich (DMS)

In einem DMS-Tabellenbereich (DMS - Database Managed Space) steuert der Datenbankmanager den Speicherbereich. Das Speichermodell besteht aus einer begrenzten Anzahl von Einheiten oder Dateien, deren Speicherbereich von DB2® Universal Database (DB2 UDB) verwaltet wird. Der Datenbankadministrator entscheidet, welche Einheiten und Dateien zu verwenden sind, und DB2 UDB verwaltet den Speicherbereich dieser Einheiten und Dateien. Der Tabellenbereich ist im Wesentlichen die Implementierung eines Dateisystems, das einem bestimmten Zweck dient und dazu entwickelt wurde, die Anforderungen des Datenbankmanagers optimal zu erfüllen.

Ein DMS-Tabellenbereich, der benutzerdefinierte Tabellen und Daten enthält, kann wie folgt definiert werden:

- Als *regulärer* Tabellenbereich zum Speichern von beliebigen Tabellendaten und, optional, Indexdaten
- Als *großer* Tabellenbereich (LARGE) zum Speichern von Langfelddaten (LONG), LOB-Daten (Large Object) oder Indexdaten

Beachten Sie beim Entwerfen Ihrer DMS-Tabellenbereiche und Behälter Folgendes:

- Der Datenbankmanager arbeitet mit einheitenübergreifendem Lesen und Schreiben von Daten (Striping), um eine gleichmäßige Verteilung von Daten auf alle Behälter sicherzustellen.
- Die Maximalgröße regulärer Tabellenbereiche beträgt 64 GB für 4-KB-Seiten, 128 GB für 8-KB-Seiten, 256 GB für 16-KB-Seiten und 512 GB für 32-KB-Seiten. Die Maximalgröße großer Tabellenbereiche (LARGE) beträgt 2 TB.
- Im Unterschied zu SMS-Tabellenbereichen müssen die Behälter, die einen DMS-Tabellenbereich bilden, nicht die gleiche Größe haben. Dies wird im Normalfall jedoch nicht empfohlen, da es zu ungleichmäßiger Verteilung (Striping) auf die Behälter und nicht optimaler Leistung führt. Wenn ein Behälter voll ist, wird in DMS-Tabellenbereichen jeder verfügbare freie Speicherbereich anderer Behälter genutzt.
- Da der Speicherbereich vorab zugeordnet wird, muss er zur Verfügung stehen, bevor der Tabellenbereich erstellt werden kann. Bei Verwendung von Einheitenbehältern muss die Einheit ebenfalls mit genügend Speicherbereich für die Definition des Behälters verfügbar sein. Auf jeder Einheit kann nur ein Behälter definiert werden. Um eine Verschwendung von Speicherbereich zu vermeiden, sollten die Größe der Einheit und die Größe des Behälters äquivalent sein.

Wenn z. B. die Einheit mit 5 000 Seiten zugeordnet ist, und der Einheitenbehälter zum Zuordnen von 3 000 Seiten definiert ist, sind 2 000 Seiten der Einheit nicht verwendbar.

- Standardmäßig wird ein EXTENTSIZE großer Speicherbereich in jedem Behälter für Systemaufwand reserviert. Nur ganze, durch EXTENTSIZE definierte Speicherbereiche werden verwendet. Für eine optimale Speicherverwaltung können Sie daher beim Zuordnen eines Behälters eine geeignete Größe anhand der folgenden Formel bestimmen:

  ```
  extent_size * (n + 1)
  ```

  Dabei ist *extent_size* die Größe jedes durch EXTENTSIZE definierten Speicherbereichs im Tabellenbereich, und *n* ist die Anzahl dieser Speicherbereiche, die Sie in dem Behälter speichern wollen.
- Die Mindestgröße eines DMS-Tabellenbereichs beträgt fünf EXTENTSIZE große Speicherbereiche. Jeder Versuch, einen Tabellenbereich zu erstellen, der kleiner als fünf EXTENTSIZE-Bereiche ist, führt zu einem Fehler (SQL1422N).
  - Drei EXTENTSIZE große Speicherbereiche im Tabellenbereich sind für den Systemaufwand reserviert.
  - Mindestens zwei EXTENTSIZE große Speicherbereiche sind erforderlich, um Tabellendaten des Benutzers zu speichern. (Diese zwei Speicherbereiche sind für die regulären Daten einer Tabelle vorgesehen, und nicht für Index-, Langfeld- oder LOB-Daten, die eigene Speicherbereiche benötigen.)
- Einheitenbehälter müssen logische Datenträger mit einer „zeichenspezifischen Schnittstelle" (d. h. keine physischen Datenträger) verwenden.
- Für DMS-Tabellenbereiche können auch Dateien anstelle von Einheiten (devices) verwendet werden. Zwischen einer Datei und einer Einheit gibt es keine betriebsbedingten Unterschiede. Jedoch kann eine Datei wegen des zur Laufzeit mit dem Dateisystem verbundenen Systemaufwands weniger effizient sein. Dateien sind in folgenden Situationen nützlich:
  - Wenn Einheiten nicht direkt unterstützt werden.
  - Wenn eine Einheit nicht verfügbar ist.
  - Wenn die Maximalleistung nicht erforderlich ist.
  - Wenn Sie keine Einheiten einrichten wollen.
- Wenn LOB- oder LONG VARCHAR-Daten verarbeitet werden, kann die Verwendung des Dateisystemcache Leistungsvorteile erbringen. Beachten Sie, dass LOB- und LONG VARCHAR-Daten nicht vom DB2 UDB-Pufferpool zwischengespeichert (gepuffert) werden.
- Einige Betriebssysteme erlauben den Betrieb physischer Einheiten, die größer als 2 GB sind. Sie sollten in Betracht ziehen, die physische Einheit in logische Einheiten zu partitionieren, damit kein Behälter größer als die vom Betriebssystem zugelassene Größe ist.

Zugehörige Konzepte:

- „Aufbau von Tabellenbereichen" auf Seite 100
- „Vom Betriebssystem verwalteter Speicherbereich (SMS)" auf Seite 104
- „SMS- und DMS-Tabellenbereiche im Vergleich" auf Seite 125
- „Tabellenbereichszuordnungen" auf Seite 108
- „Hinzufügen und Erweitern von Behältern in DMS-Tabellenbereichen" auf Seite 112

Tabellenbereichszuordnungen

Eine Tabellenbereichszuordnung ist eine interne Darstellung in DB2® Universal Database (DB2 UDB) eines DMS-Tabellenbereichs, die die Konvertierung logischer Seitenpositionen in physische Seitenpositionen in einem Tabellenbereich beschreibt. Die folgenden Informationen erläutern die Nützlichkeit einer Tabellenbereichszuordnung und die Herkunft der Informationen in einer Tabellenbereichszuordnung.

In einer DB2 UDB-Datenbank werden Seiten in einem DMS-Tabellenbereich logisch von 0 bis (N-1) durchnummeriert, wobei N die Anzahl verwendbarer Seiten im Tabellenbereich darstellt.

Die Seiten in einem Tabellenbereich werden zu Speicherbereichen gruppiert, deren Größe durch den Parameter EXTENTSIZE definiert ist. Aus Sicht der Tabellenbereichsverwaltung erfolgt jede Zuordnung von Objekten auf der Grundlage der durch EXTENTSIZE definierten Speicherbereiche. Auf diese Weise ist es möglich, dass eine Tabelle vielleicht nur die Hälfte der Seiten in einem EXTENTSIZE-Speicherbereich verwendet, jedoch wird der gesamte Speicherbereich als in Gebrauch und dem jeweiligen Objekt zugeordnet betrachtet. Standardmäßig wird ein EXTENTSIZE großer Speicherbereich zur Aufnahme der Behälterkennung verwendet, wobei dieser Speicherbereich nicht zum Speichern von Daten verwendet werden kann. Wenn jedoch die Registriervariable DB2_USE_PAGE_CONTAINER_TAG aktiviert ist, wird nur eine Seite für die Behälterkennung verwendet.

Da der Speicher in Behältern jeweils in Form eines EXTENTSIZE großen Bereichs zugeordnet wird, werden Seiten, die nicht eine volle EXTENTSIZE-Größe bilden, nicht verwendet. Wenn Sie zum Beispiel einen 205 Seiten großen Behälter mit einem EXTENTSIZE-Wert 10 haben, sind ein EXTENTSIZE-Speicherbereich für die Kennung und 19 EXTENTSIZE-Speicherbereiche für Daten verfügbar. Die fünf übrigen Seiten werden verschenkt.

Wenn ein DMS-Tabellenbereich einen einzelnen Behälter enthält, ist die Umwandlung von der logischen Seitennummer in die physische Position auf dem Datenträger ein einfacher Prozess, bei dem die Seiten 0, 1, 2 in der gleichen Reihenfolge auf dem Datenträger angeordnet werden.

Ebenfalls recht einfach ist der Prozess, wenn mehr als ein Behälter vorhanden ist und jeder der Behälter gleich groß ist. Der erste EXTENTSIZE-Speicherbereich im Tabellenbereich (der die Seiten 0 bis (extentsize - 1)) enthält, wird im ersten Behälter angelegt, der zweite EXTENTSIZE-Speicherbereich wird im zweiten Behälter angelegt usw. Nach dem letzten Behälter wird der Prozess in einem Reihumverfahren wiederholt, wobei wieder mit dem ersten Behälter begonnen wird.

Für Tabellenbereiche, die Behälter unterschiedlicher Größen enthalten, kann kein einfaches Reihumverfahren angewandt werden, da in diesem Fall der zusätzliche Speicherplatz in den größeren Behältern nicht genutzt wird. An dieser Stelle kommt die Tabellenbereichszuordnung (engl. table space map) ins Spiel: Sie gibt an, wie die EXTENTSIZE-Speicherbereiche innerhalb des Tabellenbereichs positioniert sind, und stellt dadurch sicher, dass alle Speicherbereiche in den physischen Behältern zur Verwendung verfügbar sind.

Anmerkung: In den folgenden Beispielen wird bei den Behältergrößen die Größe der Behälterkennung (container tag) nicht berücksichtigt. Die Behältergrößen sind sehr klein und dienen lediglich zu Veranschaulichungszwecken. Sie stellen keine empfohlenen Behältergrößen dar. Die Beispiele zeigen Behälter unterschiedlicher Größen innerhalb eines Tabellenbereichs, jedoch wird empfohlen, Behälter gleicher Größe zu verwenden.

Beispiel 1:

In einem Tabellenbereich sind drei Behälter vorhanden, jeder Behälter enthält 80 verwendbare Seiten und der EXTENTSIZE-Wert für den Tabellenbereich beträgt 20. Jeder Behälter enthält daher vier EXTENTSIZE große Speicherbereiche (80 / 20) mit insgesamt zwölf Speicherbereichen. Diese Speicherbereiche (Extents) befinden sich auf dem Datenträger wie in Abb. 29 zu sehen ist.

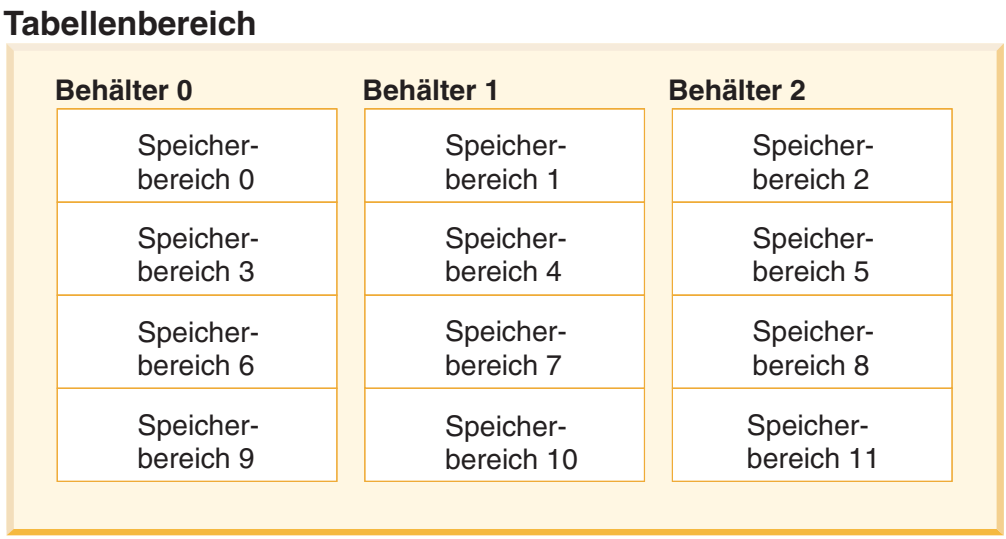


Abbildung 29. Tabellenbereich mit drei Behältern und zwölf Speicherbereichen

Wenn Sie sich eine Tabellenbereichszuordnung ansehen wollen, erstellen Sie mit Hilfe des Snapshot Monitor eine Momentaufnahme des Tabellenbereichs. In Beispiel 1, in dem die drei Behälter die gleiche Größe besitzen, sieht die Tabellenbereichszuordnung wie folgt aus:

```
Range     Stripe  Stripe  Max        Max    Start    End    Adj.   Containers
Number    Set     Offset  Extent     Page   Stripe   Stripe
   [0]       [0]       0      11      239        0        3    0   3 (0, 1, 2)
```

Ein *Bereich* (engl. *range*) ist der Teil der Zuordnung, in dem ein zusammenhängender Bereich von Stripes jeweils die gleiche Gruppe von Behältern enthalten. In Beispiel 1 enthalten alle Stripes (0 - 3) die gleiche Gruppe von drei Behältern (0, 1 und 2), so dass dies als ein einzelner Bereich betrachtet wird.

Die Spaltenüberschriften in der Tabellenbereichszuordnung heißen 'Range Number' (Bereichsnummer), 'Stripe Set', 'Stripe Offset', 'Maximum extent number addressed by the range' (höchste Speicherbereichsnummer, die durch den Bereich adressiert wird), 'Maximum page number addressed by the range' (höchste Seitennummer, die durch den Bereich adressiert wird), 'Start Stripe' (Anfangsstripe), 'End Stripe' (Endstripe), 'Range adjustment' (Bereichsanpassung) und 'Container list' (Behälterliste). Diese Namen werden in Beispiel 2 detaillierter beschrieben.

Dieser Tabellenbereich kann auch wie in Abb. 30 dargestellt werden, wobei jede vertikale Linie einem Behälter entspricht und jede horizontale Linie als *Stripe* (einheitenübergreifend gespeicherter Datenblock) bezeichnet wird. Jede Zellennummer entspricht einem EXTENTSIZE großen Speicherbereich (Extent).

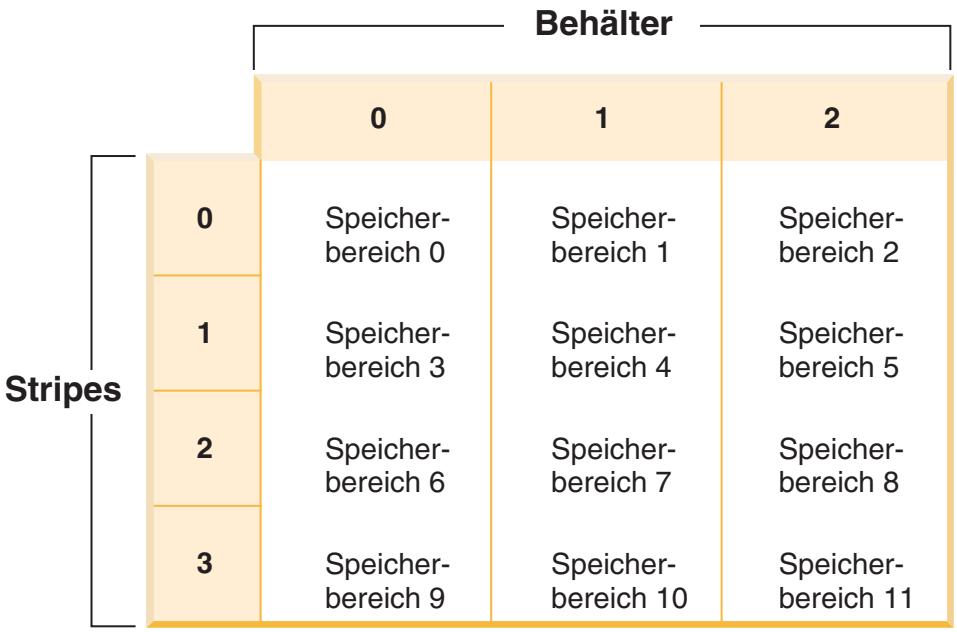


Abbildung 30. Tabellenbereich mit drei Behältern und zwölf Speicherbereichen, Stripes hervorgehoben

Beispiel 2:

Im Tabellenbereich sind zwei Behälter vorhanden: der erste ist 100 Seiten groß, der zweite 50 Seiten, und EXTENTSIZE definiert eine Größe von 25 Seiten. Dies bedeutet, dass der erste Behälter vier EXTENTSIZE große Speicherbereiche und der zweite Behälter zwei solche Speicherbereiche besitzt. Der Tabellenbereich lässt sich wie in Abb. 31 auf Seite 111 darstellen.

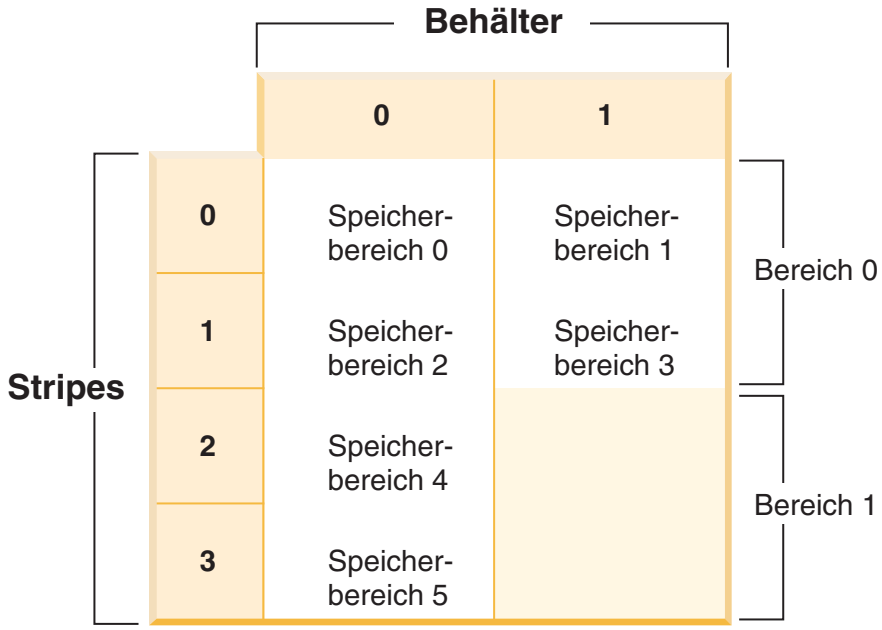


Abbildung 31. Tabellenbereich mit zwei Behältern, Bereiche hervorgehoben

Stripes 0 und 1 enthalten beide Behälter (0 und 1), jedoch enthalten die Stripes 2 und 3 nur den ersten Behälter (0). Jede dieser Gruppen von Stripes wird als Bereich (range) bezeichnet. Die Tabellenbereichszuordnung, die in der Momentaufnahme eines Tabellenbereichs gezeigt wird, sieht folgendermaßen aus:

```
Range     Stripe  Stripe  Max       Max    Start    End    Adj.   Containers
Number    Set     Offset  Extent    Page   Stripe   Stripe
   [0]       [0]       0       3      99        0       1    0   2 (0, 1)
   [1]       [0]       0       5     149        2       3    0   1 (0)
```

Im ersten Bereich (Range) befinden sich vier EXTENTSIZE große Speicherbereiche (Extents). Daher ist 3 die höchste Speicherbereichsnummer (Max Extent), die in diesem Bereich adressiert wird. Jeder Speicherbereich ist 25 Seiten groß, so dass sich im ersten Bereich 100 Seiten befinden. Da die Seitennummerierung ebenfalls bei 0 beginnt, ist 99 die höchste Seitennummer (Max Page), die in diesem Bereich adressiert wird. Der erste Stripe (Start Stripe) in diesem Bereich ist 0 und der letzte Stripe (End Stripe) im Bereich ist Stripe 1. In diesem Bereich befinden sich zwei Behälter, die die Nummern 0 und 1 haben. Das Stripe-Offset ist der erste Stripe im Stripeset, in diesem Fall 0, weil nur ein Stripset vorhanden ist. Die Bereichsanpassung (Adj.) ist ein Offsetwert, der verwendet wird, wenn Daten in einem Tabellenbereich neu ausgeglichen werden. (Ein Neuausgleich kann stattfinden, wenn in einem Tabellenbereich Speicherplatz hinzugefügt oder gelöscht wird.) Wenn kein Neuausgleich stattfindet, ist dieser Wert immer 0.

Im zweiten Bereich (Range) befinden sich zwei EXTENTSIZE große Speicherbereiche (Extents), und da 3 die höchste Speicherbereichsnummer ist, die im vorigen Bereich adressiert wurde, ist 5 die höchste Speicherbereichsnummer, die in diesem Bereich adressiert wird. Im zweiten Bereich befinden sich 50 Seiten (2 Speicherbereiche * 25 Seiten), und da 99 die höchste Seitennummer ist, die im vorigen Bereich adressiert wird, ist 149 nun die höchste Seitennummer, die in diesem Bereich adressiert wird. Dieser Bereich beginnt bei Stripe 3 und endet bei Stripe 3.

Zugehörige Konzepte:

- „Von der Datenbank verwalteter Speicherbereich (DMS)" auf Seite 106
- „Snapshot monitor" in *System Monitor Guide and Reference*
- „Hinzufügen und Erweitern von Behältern in DMS-Tabellenbereichen" auf Seite 112
- „Löschen und Verkleinern von Behältern in DMS-Tabellenbereichen" auf Seite 122

Zugehörige Referenzen:

- „GET SNAPSHOT Command" in *Command Reference*

Hinzufügen und Erweitern von Behältern in DMS-Tabellenbereichen

Bei der Erstellung eines Tabellenbereichs wird die zugehörige Tabellenbereichszuordnung erstellt und alle Anfangsbehälter werden so ausgerichtet, dass sie in Stripe 0 beginnen (Stripe - einheitenübergreifend gespeicherter Datenblock). Dies bedeutet, dass die Daten gleichmäßig über sämtliche Tabellenbereichsbehälter verteilt gespeichert werden, bis die einzelnen Behälter gefüllt sind. (Siehe „Beispiel 1" auf Seite 114.)

Die Anweisung ALTER TABLESPACE gibt Ihnen die Möglichkeit, einem vorhandenen Tabellenbereich einen Behälter hinzuzufügen oder einen Behältern zu vergrößern, um seine Speicherkapazität zu erhöhen.

Durch das Hinzufügen eines Behälters, der kleiner als vorhandene Behälter ist, wird eine ungleichmäßige Verteilung der Daten verursacht. Dies kann dazu führen, dass parallele Operationen wie das Vorablesen von Daten weniger effizient arbeiten, als sie es bei Behältern gleicher Größe könnten.

Wenn einem Tabellenbereich neue Behälter hinzugefügt oder vorhandene Behälter erweitert werden, *kann* ein Neuausgleich der Tabellenbereichsdaten stattfinden.

Neuausgleich

Der Prozess des Neuausgleichs beim Hinzufügen oder Erweitern von Behältern besteht im Versetzen von EXTENTSIZE großen Speicherbereichen des Tabellenbereichs von einer Position an eine andere. Dies geschieht, um die einheitenübergreifende Speicherung der Daten innerhalb des Tabellenbereichs beizubehalten.

Während dieser Neuverteilung der Daten wird der Zugriff auf den Tabellenbereich nicht eingeschränkt. Objekte können wie gewöhnlich gelöscht, erstellt, mit Daten gefüllt und abgefragt werden. Allerdings kann die Operation des Datenneuausgleichs erhebliche Auswirkungen auf die Leistung haben. Wenn mehr als ein Behälter hinzugefügt werden muss und Sie planen, die Behälter neu auszugleichen, sollten diese Behälter gleichzeitig innerhalb einer einzigen Anweisung ALTER TABLESPACE hinzugefügt werden, um zu vermeiden, dass der Datenbankmanager die Daten mehr als einmal neu verteilen muss.

Die obere Grenze des Tabellenbereichs spielt eine Schlüsselrolle im Neuausgleichsprozess. Die obere Grenze ist die Seitennummer der höchsten zugeordneten Seite im Tabellenbereich. Zum Beispiel hat ein Tabellenbereich 1000 Seiten und einen EXTENTSIZE-Wert von 10, was 100 EXTENTSIZE großen Speicherbereichen entspricht. Wenn der 42ste Speicherbereich der höchste zugeordnete Speicherbereich im Tabellenbereich ist, bedeutet dies, dass die obere Grenze bei 42 * 10 = 420 Seiten liegt. Dieser Wert ist nicht identisch mit dem Wert für verwendete Seiten, weil einige der Speicherbereiche unterhalb der oberen Grenze freigegeben worden sein könnten, so dass sie zur Wiederverwendung verfügbar sind.

Bevor der Neuausgleich startet, wird auf der Grundlage der durchgeführten Behälteränderungen eine neue Tabellenbereichszuordnung generiert. Die Neuausgleichsfunktion versetzt EXTENTSIZE große Speicherbereiche von ihrer Position, die durch die aktuelle Zuordnung festgelegt ist, an die Position, die durch die neue Zuordnung festgelegt wird. Die Neuausgleichsfunktion beginnt mit dem Speicherbereich 0 und versetzt jeweils einen Speicherbereich gleichzeitig, bis der Speicherbereich, der die obere Grenze enthält, versetzt wurde. Beim Versetzen der einzelnen Speicherbereiche wird die aktuelle Zuordnung stückweise in das Aussehen der neuen Zuordnung geändert. An dem Punkt, an dem der Neuausgleich abgeschlossen ist, sollten die aktuelle Zuordnung und die neue Zuordnung bis zu dem Stripe identisch aussehen, der die obere Grenze enthält. Die aktuelle Zuordnung wird dann vollständig an das Aussehen der neuen Zuordnung angeglichen und der Neuausgleichsprozess ist abgeschlossen. Wenn die Position eines Speicherbereichs in der aktuellen Zuordnung mit seiner Position in der neuen Zuordnung übereinstimmt, wird der Speicherbereich nicht versetzt, und es finden keine E/A-Operationen statt.

Wenn ein neuer Behälter hinzugefügt wird, hängt die Positionierung dieses Behälters innerhalb der neuen Zuordnung von seiner Größe sowie der Größe der anderen Behälter in seinem Stripeset ab. Wenn der Behälter groß genug ist, um im ersten Stripe des Stripeset zu beginnen und im letzten Stripe (oder dahinter) des Stripeset zu enden, wird er in dieser Weise angeordnet (siehe „Beispiel 2" auf Seite 115). Wenn der Behälter dazu nicht groß genug ist, wird er in der Zuordnung so positioniert, dass er im letzten Stripe des Stripeset endet (siehe „Beispiel 4" auf Seite 118.) Dies geschieht, um das Volumen der Daten zu minimieren, die neu ausgeglichen werden müssen.

Anmerkung: In den folgenden Beispielen wird bei den Behältergrößen die Größe der Behälterkennung (container tag) nicht berücksichtigt. Die Behältergrößen sind sehr klein und dienen lediglich zu Veranschaulichungszwecken. Sie stellen keine empfohlenen Behältergrößen dar. Die Beispiele zeigen Behälter unterschiedlicher Größen innerhalb eines Tabellenbereichs, jedoch wird empfohlen, Behälter gleicher Größe zu verwenden.

Beispiel 1:

Wenn Sie einen Tabellenbereich mit drei Behältern und einem EXTENTSIZE-Wert von 10 erstellen, und die Behälter 60, 40 bzw. 80 Seiten (3, 4 und 8 EXTENTSIZE-Werte) groß sind, wird der Tabellenbereich mit einer Zuordnung erstellt, die wie in Abb. 32 dargestellt werden kann.

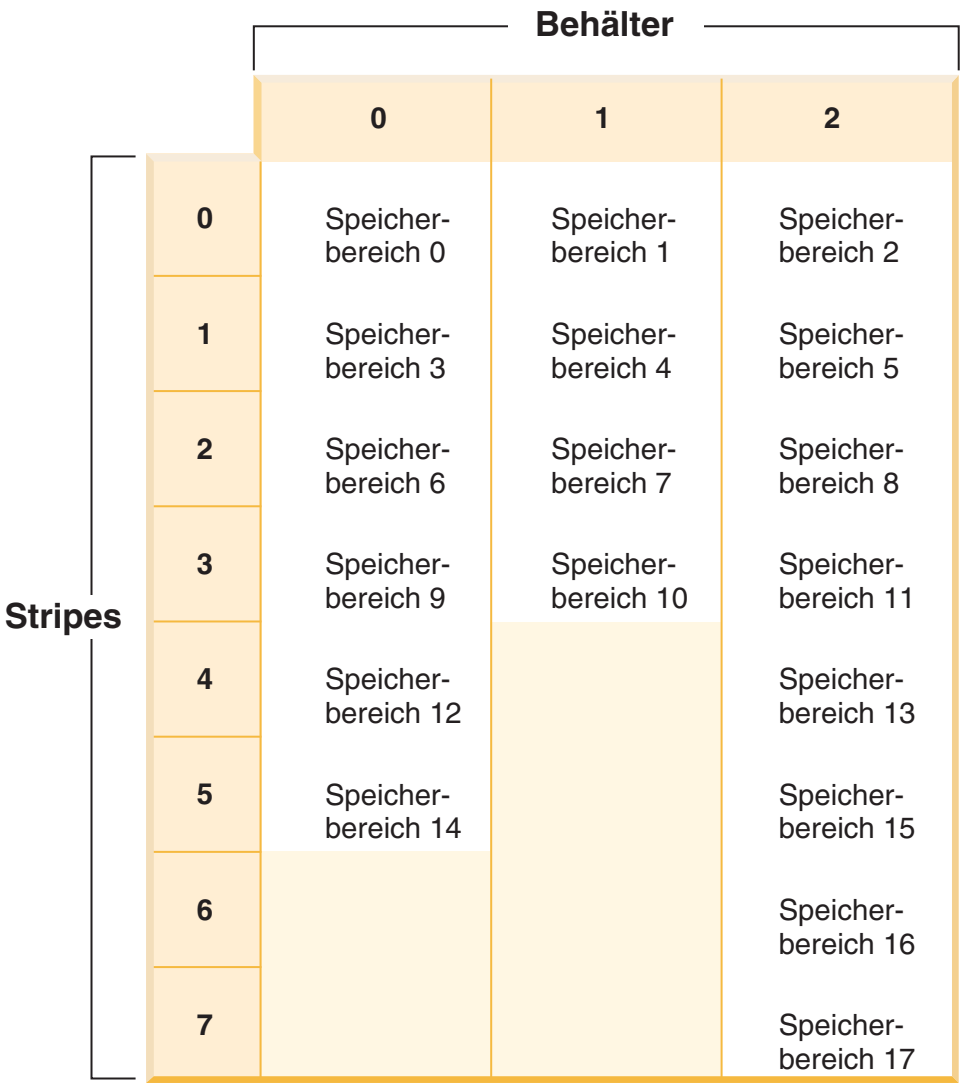


Abbildung 32. Tabellenbereich mit drei Behältern und 18 Speicherbereichen

Die entsprechende Tabellenbereichszuordnung, wie sie in einer Momentaufnahme des Tabellenbereichs gezeigt würde, sieht folgendermaßen aus:

```
Range     Stripe  Stripe  Max        Max    Start    End    Adj.   Containers
Number    Set     Offset  Extent     Page   Stripe   Stripe
   [0]      [0]       0      11      119        0       3    0   3 (0, 1, 2)
   [1]      [0]       0      15      159        4       5    0   2 (0, 2)
   [2]      [0]       0      17      179        6       7    0   1 (2)
```

Die Spaltenüberschriften in der Tabellenbereichszuordnung heißen 'Range Number' (Bereichsnummer), 'Stripe Set', 'Stripe Offset', 'Maximum extent number addressed by the range' (höchste Speicherbereichsnummer, die durch den Bereich adressiert wird), 'Maximum page number addressed by the range' (höchste Seitennummer, die durch den Bereich adressiert wird), 'Start Stripe' (Anfangsstripe), 'End Stripe' (Endstripe), 'Range adjustment' (Bereichsanpassung) und 'Container list' (Behälterliste).

Beispiel 2:

Wenn dem Tabellenbereich in Beispiel 1 ein 80 Seiten großer Behälter hinzugefügt wird, ist der Behälter groß genug, um im ersten Stripe (Stripe 0) zu beginnen und im letzten Stripe (Stripe 7) zu enden. Er wird so positioniert, dass er im ersten Stripe beginnt. Der resultierende Tabellenbereich lässt sich wie in Abb. 33 darstellen.

		Behälter			
		0	1	2	3
Stripes	0	Speicherbereich 0	Speicherbereich 1	Speicherbereich 2	Speicherbereich 3
	1	Speicherbereich 4	Speicherbereich 5	Speicherbereich 6	Speicherbereich 7
	2	Speicherbereich 8	Speicherbereich 9	Speicherbereich 10	Speicherbereich 11
	3	Speicherbereich 12	Speicherbereich 13	Speicherbereich 14	Speicherbereich 15
	4	Speicherbereich 16		Speicherbereich 17	Speicherbereich 18
	5	Speicherbereich 19		Speicherbereich 20	Speicherbereich 21
	6			Speicherbereich 22	Speicherbereich 23
	7			Speicherbereich 24	Speicherbereich 25

Abbildung 33. Tabellenbereich mit vier Behältern und 26 Speicherbereichen

Die entsprechende Tabellenbereichszuordnung, wie sie in einer Momentaufnahme des Tabellenbereichs gezeigt würde, sieht folgendermaßen aus:

```
Range     Stripe  Stripe  Max        Max    Start    End    Adj.   Containers
Number    Set     Offset  Extent     Page   Stripe   Stripe
   [0]       [0]       0      15      159        0       3    0   4 (0, 1, 2, 3)
   [1]       [0]       0      21      219        4       5    0   3 (0, 2, 3)
   [2]       [0]       0      25      259        6       7    0   2 (2, 3)
```

Wenn sich die obere Grenze im Speicherbereich 14 befindet, startet die Neuausgleichsfunktion bei Speicherbereich 0 und versetzt alle Speicherbereiche bis einschließlich Speicherbereich 14. Die Position von Speicherbereich 0 innerhalb der beiden Zuordnungen ist identisch, so dass dieser Speicherbereich nicht versetzt werden muss. Das Gleiche gilt für die Speicherbereiche 1 und 2. Speicherbereich 3 muss versetzt werden, so dass der Speicherbereich von der alten Position (zweiter Speicherbereich in Behälter 0) gelesen und an die neue Position (erster Speicherbereich in Behälter 3) geschrieben wird. Jeder Speicherbereich nach diesem bis einschließlich Speicherbereich 14 wird versetzt. Wenn Speicherbereich 14 versetzt wurde, sieht die aktuelle Zuordnung wie die neue Zuordnung aus, und die Neuausgleichsfunktion wird beendet.

Wenn die Zuordnung so geändert wird, dass sämtlicher neu hinzugefügter Speicherplatz über der oberen Grenze liegt, ist kein Neuausgleich erforderlich und der gesamte Speicherplatz ist sofort zur Verwendung verfügbar. Wenn die Zuordnung so geändert wird, dass einiger Speicherplatz oberhalb der oberen Grenze liegt, ist der Speicherplatz in den Stripes oberhalb der oberen Grenze verfügbar. Der übrige Speicherplatz ist erst verfügbar, wenn die Neuausgleichsfunktion abgeschlossen ist.

Wenn Sie einen Behälter erweitern, arbeitet die Funktion des Neuausgleichs ähnlich. Wenn ein Behälter so erweitert wird, dass er über den letzten Stripe in seinem Stripeset hinausreicht, wird das Stripeset entsprechend vergrößert, und die folgenden Stripesets werden entsprechend nach hinten verschoben. Infolgedessen reicht der Behälter nicht in eines der nachfolgenden Stripesets hinein.

Beispiel 3:

Betrachten Sie den Tabellenbereich aus Beispiel 1. Wenn Sie den Behälter 1 von 40 Seiten auf 80 Seiten erweitern, sieht der neue Tabellenbereich wie in Abb. 34 aus.

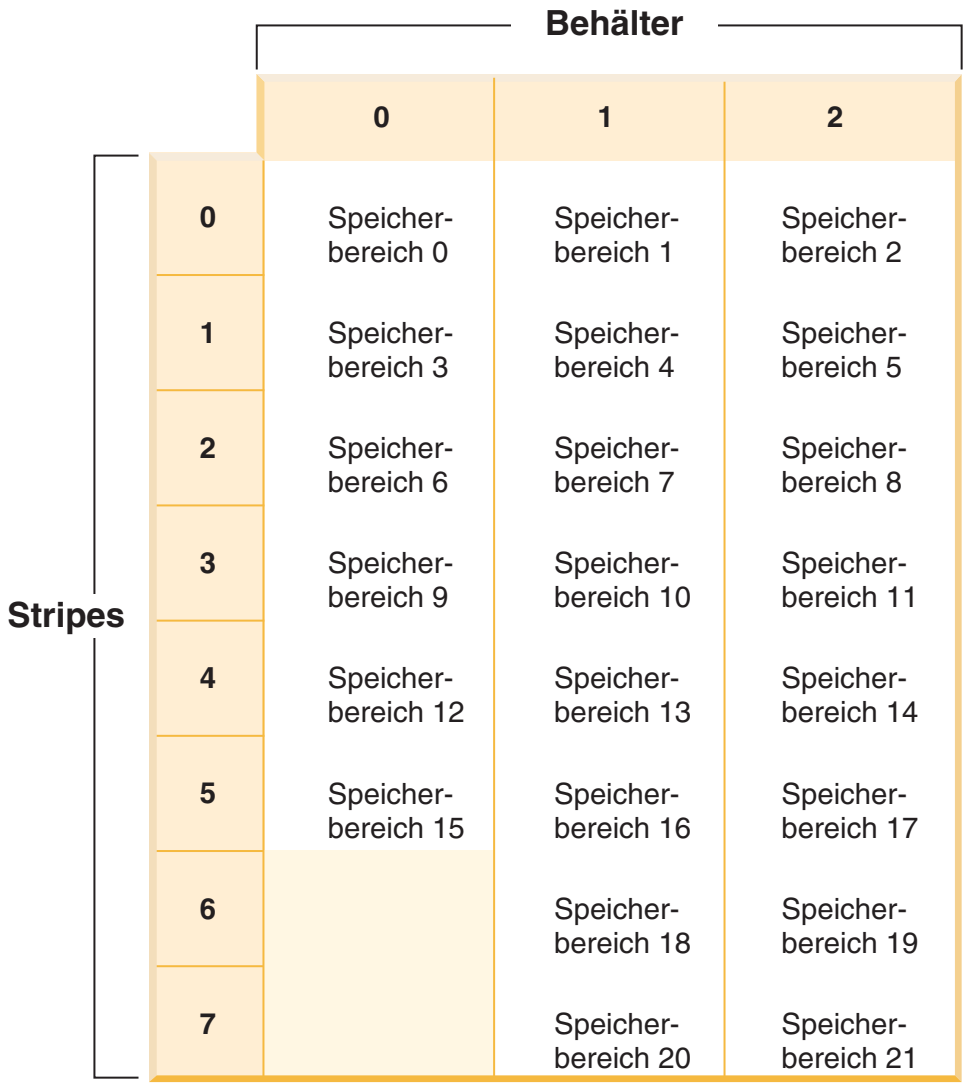


Abbildung 34. Tabellenbereich mit drei Behältern und 22 Speicherbereichen

Die entsprechende Tabellenbereichszuordnung, wie sie in einer Momentaufnahme des Tabellenbereichs gezeigt würde, sieht folgendermaßen aus:

```
Range     Stripe  Stripe  Max        Max    Start    End    Adj.   Containers
Number    Set     Offset  Extent     Page   Stripe   Stripe
   [0]       [0]       0      17      179        0       5    0   3 (0, 1, 2)
   [1]       [0]       0      21      219        6       7    0   2 (1, 2)
```

Beispiel 4:

Betrachten Sie den Tabellenbereich aus „Beispiel 1" auf Seite 114. Wenn ein 50 Seiten großer Behälter (fünf EXTENTSIZE-Größen) hinzugefügt wird, wird der Behälter der neuen Zuordnung wie folgt hinzugefügt. Der Behälter ist nicht groß genug, um im ersten Stripe (Stripe 0) zu beginnen und im letzten Stripe (Stripe 7) oder dahinter zu enden. Daher wird er in der Weise angeordnet, dass er im letzten Stripe endet. (Siehe Abb. 35.)

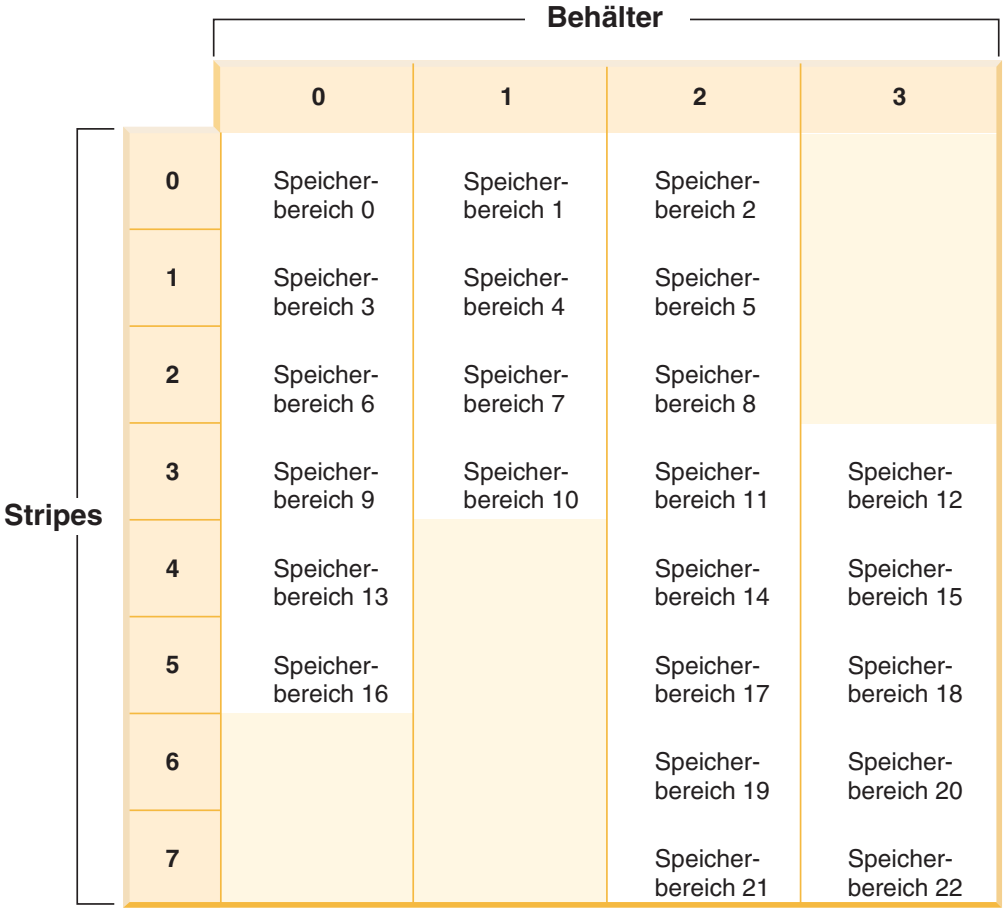


Abbildung 35. Tabellenbereich mit vier Behältern und 23 Speicherbereichen

Die entsprechende Tabellenbereichszuordnung, wie sie in einer Momentaufnahme des Tabellenbereichs gezeigt würde, sieht folgendermaßen aus:

```
Range     Stripe  Stripe  Max       Max    Start    End    Adj.   Containers
Number    Set     Offset  Extent    Page   Stripe   Stripe
   [0]      [0]       0       8      89        0       2    0   3 (0, 1, 2)
   [1]      [0]       0      12     129        3       3    0   4 (0, 1, 2, 3)
   [2]      [0]       0      18     189        4       5    0   3 (0, 2, 3)
   [3]      [0]       0      22     229        6       7    0   2 (2, 3)
```

Verwenden Sie zur Erweiterung eines Behälters die Option EXTEND oder RESIZE in der Anweisung ALTER TABLESPACE. Zum Hinzufügen eines Behälters und Neuausgleichen der Daten dient die Option ADD in der Anweisung ALTER TABLESPACE. Wenn Sie einem Tabellenbereich, der bereits mehr als ein Stripeset hat, einen Behälter hinzufügen, können Sie angeben, welchem Stripeset der Behälter hinzugefügt werden soll. Dazu verwenden Sie die Option ADD TO STRIPE SET in der Anweisung ALTER TABLESPACE. Wenn Sie kein Stripeset angeben, wird der Behälter standardmäßig dem aktuellen Stripeset hinzugefügt. Das aktuelle Stripeset ist das zuletzt erstellte Stripeset, nicht das, dem zuletzt Speicherplatz hinzugefügt wurde.

Jede Änderung an einem Stripeset kann einen Neuausgleich in diesem Stripeset und den anderen nachfolgenden Stripesets zur Folge haben.

Sie können den Fortschritt eines Neuausgleichs mit Hilfe von Momentaufnahmen der Tabellenbereiche überwachen. Eine Momentaufnahme eines Tabellenbereichs kann Informationen über einen Neuausgleich liefern, wie zum Beispiel den Startzeitpunkt des Neuausgleichs, die Anzahl der versetzten Speicherbereiche und die Anzahl der noch zu versetzenden Speicherbereiche.

Ohne Neuausgleich (bei Verwendung von Stripesets)

Wenn Sie einen Behälter hinzufügen oder erweitern und der Speicherplatz oberhalb der oberen Grenze des Tabellenbereichs hinzugefügt wird, findet kein Neuausgleich statt.

Durch Hinzufügen eines Behälters wird beinahe immer Speicherplatz unterhalb der oberen Grenze hinzugefügt. Mit anderen Worten, wenn Sie einen Behälter hinzufügen, ist häufig ein Neuausgleich erforderlich. Es ist eine Option verfügbar, mit der verlasst wird, dass neue Behälter oberhalb der oberen Grenze hinzugefügt werden. Dadurch ergibt sich die Möglichkeit, keinen Neuausgleich des Inhalts des Tabellenbereichs durchzuführen. Ein Vorteil dieser Methode ist, dass der neue Behälter sofort zur Verwendung verfügbar ist. Die Option, keinen Neuausgleich durchzuführen, gilt nur für das Hinzufügen von Behältern, nicht für die Erweiterung vorhandener Behälter. Bei der Erweiterung von Behältern lässt sich ein Neuausgleich nur vermeiden, wenn der Speicherplatz, den Sie hinzufügen, oberhalb der oberen Grenze liegt. Wenn Sie zum Beispiel eine Anzahl von Behältern haben, die die gleiche Größe besitzen, und Sie jeden von ihnen um den gleichen Betrag erweitern, ändern sich die relativen Positionen der Speicherbereiche nicht, und es findet kein Neuausgleich statt.

Das Hinzufügen von Behältern ohne Neuausgleich geschieht durch Hinzufügen eines neuen *Stripeset*. Ein Stripeset ist eine Gruppe von Behältern in einem Tabellenbereich, die Datenblöcke enthält, die einheitenübergreifend in ihr gespeichert sind, und die von den anderen Behältern getrennt ist, die zu diesem Tabellenbereich gehören. Sie verwenden ein neues Stripeset, wenn Sie Behälter einem Tabellenbereich hinzufügen wollen, ohne die Daten neu auszugleichen. Die vorhanden Behälter in den vorhandenen Stripesets bleiben unberührt, und die Behälter, die Sie hinzufügen, werden Teil eines neuen Stripeset.

Verwenden Sie die Option BEGIN NEW STRIPE SET in der Anweisung ALTER TABLESPACE, wenn Sie Behälter ohne Neuausgleich hinzufügen wollen.

Beispiel 5:

Wenn Sie einen Tabellenbereich mit drei Behältern und einem EXTENTSIZE-Wert von 10 haben, und die Behälter 30, 40 und 40 Seiten (3, 4 und 4 EXTENTSIZE-Werte) groß sind, kann der Tabellenbereich wie in Abb. 36 dargestellt werden.

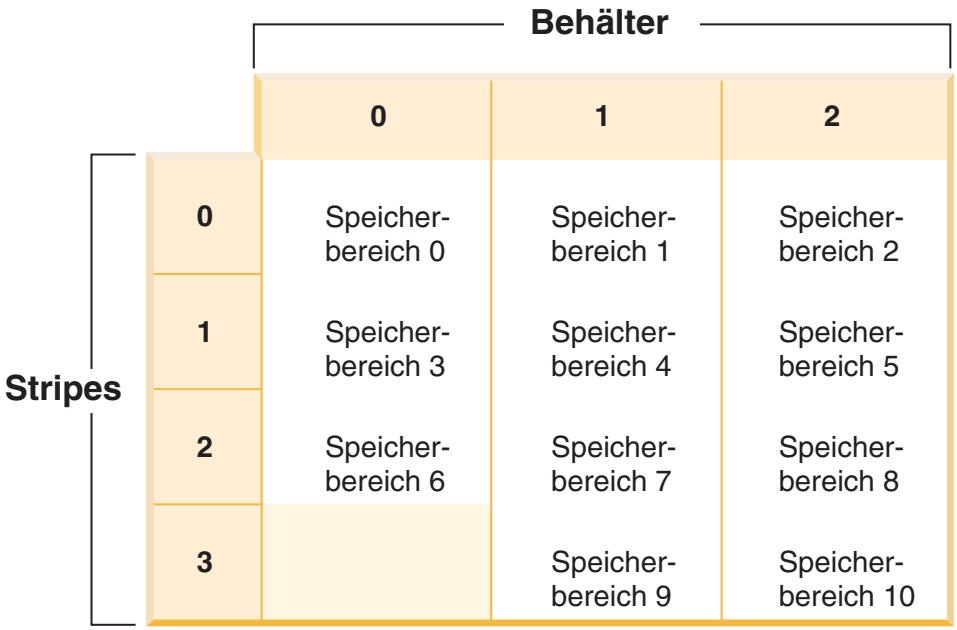


Abbildung 36. Tabellenbereich mit drei Behältern und 11 Speicherbereichen

Die entsprechende Tabellenbereichszuordnung, wie sie in einer Momentaufnahme des Tabellenbereichs gezeigt würde, sieht folgendermaßen aus:

```
Range     Stripe  Stripe  Max        Max    Start    End    Adj.   Containers
Number    Set     Offset  Extent     Page   Stripe   Stripe
   [0]       [0]       0       8       89        0       2    0   3 (0, 1, 2)
   [1]       [0]       0      10      109        3       3    0   2 (1, 2)
```

Beispiel 6:

Wenn Sie zwei neue Behälter mit der Option BEGIN NEW STRIPE SET hinzufügen, die 30 und 40 Seiten (3 und 4 EXTENTSIZE-Größen) groß sind), werden die vorhandenen Bereiche (Ranges) davon nicht berührt und stattdessen eine neue Gruppe von Bereichen erstellt. Diese neue Gruppe von Bereichen ist ein Stripeset und das zuletzt erstellte Stripeset wird als aktuelles Stripeset bezeichnet. Nach dem Hinzufügen der beiden neuen Behälter sieht der Tabellenbereich wie in Abb. 37 auf Seite 121 aus.

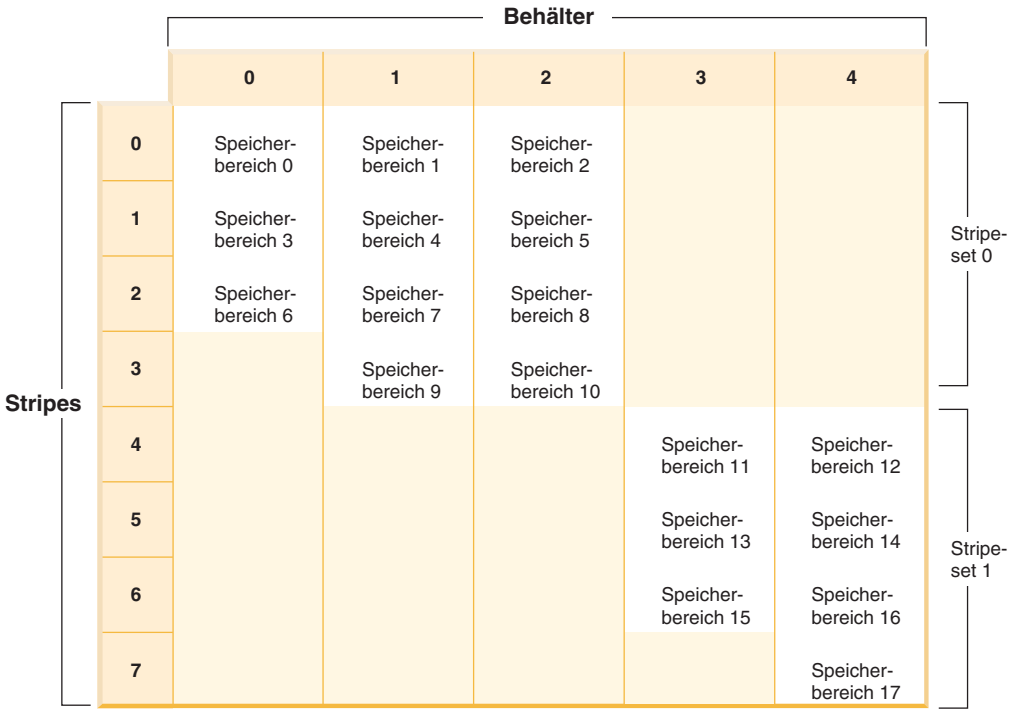


Abbildung 37. Tabellenbereich mit zwei Stripesets

Die entsprechende Tabellenbereichszuordnung, wie sie in einer Momentaufnahme des Tabellenbereichs gezeigt würde, sieht folgendermaßen aus:

```
Range     Stripe  Stripe  Max       Max    Start    End    Adj.   Containers
Number    Set     Offset  Extent    Page   Stripe   Stripe
   [0]      [0]       0       8      89        0       2    0   3 (0, 1, 2)
   [1]      [0]       0      10     109        3       3    0   2 (1, 2)
   [2]      [1]       4      16     169        4       6    0   2 (3, 4)
   [3]      [1]       4      17     179        7       7    0   1 (4)
```

Wenn Sie einem Tabellenbereich neue Behälter hinzufügen und Sie die Option TO STRIPE SET mit der Klausel ADD *nicht* verwenden, werden die Behälter dem aktuellen Stripeset (d. h. dem Stripeset mit der höchsten Nummer) hinzugefügt. Sie können mit der Klausel ADD TO STRIPE SET Behälter jedem Stripeset im Tabellenbereich hinzufügen. Sie müssen ein gültiges Stripeset angeben.

DB2® Universal Database (DB2 UDB) verwaltet die Stripesets mit Hilfe der Tabellenbereichszuordnung. Durch das Hinzufügen von Behältern ohne Neuausgleich wächst die Zuordnung in der Regel schneller als bei Durchführung eines Neuausgleichs der Behälter. Wenn die Tabellenbereichszuordnung zu groß wird, empfangen Sie den Fehler SQL0259N, wenn Sie versuchen, weitere Behälter hinzuzufügen.

Zugehörige Konzepte:

- „Tabellenbereichszuordnungen” auf Seite 108

Zugehörige Tasks:

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

- „Modifizieren von Behältern in einem DMS-Tabellenbereich" in *Systemverwaltung: Implementierung*

Zugehörige Referenzen:

- „ALTER TABLESPACE statement" in *SQL Reference, Volume 2*
- „GET SNAPSHOT Command" in *Command Reference*
- „Table space activity monitor elements" in *System Monitor Guide and Reference*

Löschen und Verkleinern von Behältern in DMS-Tabellenbereichen

Bei einem DMS-Tabellenbereich ist es möglich, einen Behälter aus dem Tabellenbereich zu löschen oder einen Behälter zu verkleinern. Dies können Sie mit Hilfe der Anweisung ALTER TABLESPACE erreichen.

Das Löschen oder Verkleinern eines Behälters ist nur zulässig, wenn die Anzahl von EXTENTSIZE großen Speicherbereichen, die durch die Operation gelöscht werden sollen, kleiner als oder gleich der Anzahl der freien EXTENTSIZE großen Speicherbereiche oberhalb der oberen Grenze im Tabellenbereich ist. Dies ist notwendig, weil durch die Operation keine Seitennummern geändert werden können und daher alle Speicherbereiche bis zur oberen Grenze (einschließlich) an der gleichen logischen Position im Tabellenbereich verbleiben müssen. Das heißt, der resultierende Tabellenbereich muss ausreichend Platz haben, um alle Daten bis zur oberen Grenze einschließlich enthalten zu können. In dem Fall, dass nicht genügend freier Speicher verbleibt, empfangen Sie sofort nach Ausführung der Anweisung eine Fehlernachricht.

Die obere Grenze ist die Seitennummer der höchsten zugeordneten Seite im Tabellenbereich. Zum Beispiel hat ein Tabellenbereich 1000 Seiten und einen EXTENTSIZE-Wert von 10, was 100 EXTENTSIZE großen Speicherbereichen entspricht. Wenn der 42ste Speicherbereich der höchste zugeordnete Speicherbereich im Tabellenbereich ist, bedeutet dies, dass die obere Grenze bei 42 * 10 = 420 Seiten liegt. Dieser Wert ist nicht identisch mit dem Wert für verwendete Seiten, weil einige der Speicherbereiche unterhalb der oberen Grenze freigegeben worden sein könnten, so dass sie zur Wiederverwendung verfügbar sind.

Wenn Behälter gelöscht oder verkleinert werden, findet ein Neuausgleich statt, wenn sich Daten in dem Speicherbereich befinden, der aus dem Tabellenbereich gelöscht wird. Vor dem Start des Neuausgleichs wird eine neue Tabellenbereichszuordnung auf der Grundlage der vorgenommenen Behälteränderungen erstellt. Die Neuausgleichsfunktion versetzt EXTENTSIZE große Speicherbereiche von ihrer Position, die durch die aktuelle Zuordnung festgelegt ist, an die Position, die durch die neue Zuordnung festgelegt wird. Die Neuausgleichsfunktion beginnt mit dem Speicherbereich, der die obere Grenze enthält, und versetzt jeweils einen Speicherbereich gleichzeitig, bis Speicherbereich 0 versetzt wurde. Beim Versetzen der einzelnen Speicherbereiche wird die aktuelle Zuordnung stückweise in das Aussehen der neuen Zuordnung geändert. Wenn die Position eines Speicherbereichs in der aktuellen Zuordnung mit seiner Position in der neuen Zuordnung übereinstimmt, wird der Speicherbereich nicht versetzt, und es finden keine E/A-Operationen statt. Da beim Neuausgleich Speicherbereiche vom höchsten zugeordneten Speicherbereich angefangen und mit dem ersten Speicherbereich im Tabellenbereich zum Schluss versetzt werden, wird er als *umgekehrter Neuausgleich* (im Gegensatz zum *Vorwärtsneuausgleich*, der stattfindet, wenn der Tabellenbereich h dem Hinzufügen oder Erweitern von Behältern vergrößert wird) bezeichnet.

Wenn Behälter gelöscht werden, werden die verbleibenden Behälter neu durchnummeriert, so dass ihre Behälter-IDs bei 0 anfangen und sich jeweils um 1 erhöhen. Wenn alle Behälter in einem Stripeset gelöscht werden, wird das Stripeset aus der Zuordnung entfernt und alle nachfolgenden Stripesets werden nach unten verschoben und neu nummeriert, so dass keine Lücken in den Nummern des Stripeset auftreten.

Anmerkung: In den folgenden Beispielen wird bei den Behältergrößen die Größe der Behälterkennung (container tag) nicht berücksichtigt. Die Behältergrößen sind sehr klein und dienen lediglich zu Veranschaulichungszwecken. Sie stellen keine empfohlenen Behältergrößen dar. Die Beispiele zeigen Behälter unterschiedlicher Größen innerhalb eines Tabellenbereichs, jedoch dient dies nur der Veranschaulichung. Zu empfehlen ist die Verwendung von Behältern gleicher Größe.

Betrachten Sie zum Beispiel einen Tabellenbereich mit drei Behältern und einem EXTENTSIZE-Wert 10. Die Behälter sind 20, 50 und 50 Seiten (d. h. 2, 5 und 5 EXTENTSIZE-Größen groß). Der Tabellenbereich lässt sich wie in Abb. 38 darstellen.

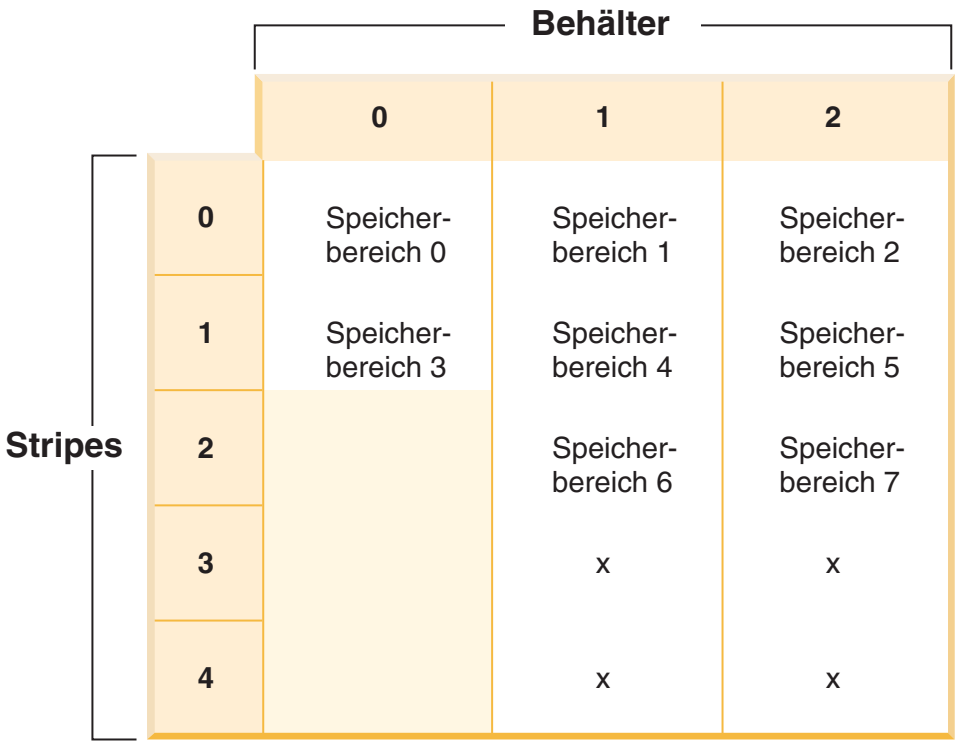


Abbildung 38. Tabellenbereich mit zwölf Speicherbereichen und vier Speicherbereichen ohne Daten

Ein X zeigt an, dass an der Stelle ein Speicherbereich vorhanden ist, der jedoch keine Daten enthält.

Wenn Sie den Behälter 0 löschen, der zwei Speicherbereiche enthält, müssen über der oberen Grenze mindestens zwei freie Speicherbereiche vorhanden sein. Die obere Grenze ist der Speicherbereich 7, so dass vier freie Speicherbereiche verbleiben. In diesem Fall können Sie also den Behälter 0 löschen.

Die entsprechende Tabellenbereichszuordnung, wie sie in einer Momentaufnahme des Tabellenbereichs gezeigt würde, sieht folgendermaßen aus:

```
Range     Stripe  Stripe  Max        Max    Start    End    Adj.   Containers
Number    Set     Offset  Extent     Page   Stripe   Stripe
   [0]       [0]       0       5       59        0       1    0   3 (0, 1, 2)
   [1]       [0]       0      11      119        2       4    0   2 (1, 2)
```

Nach dem Löschen enthält der Tabellenbereich nur Behälter 0 und Behälter 1. Der neue Tabellenbereich lässt sich wie in Abb. 39 darstellen.

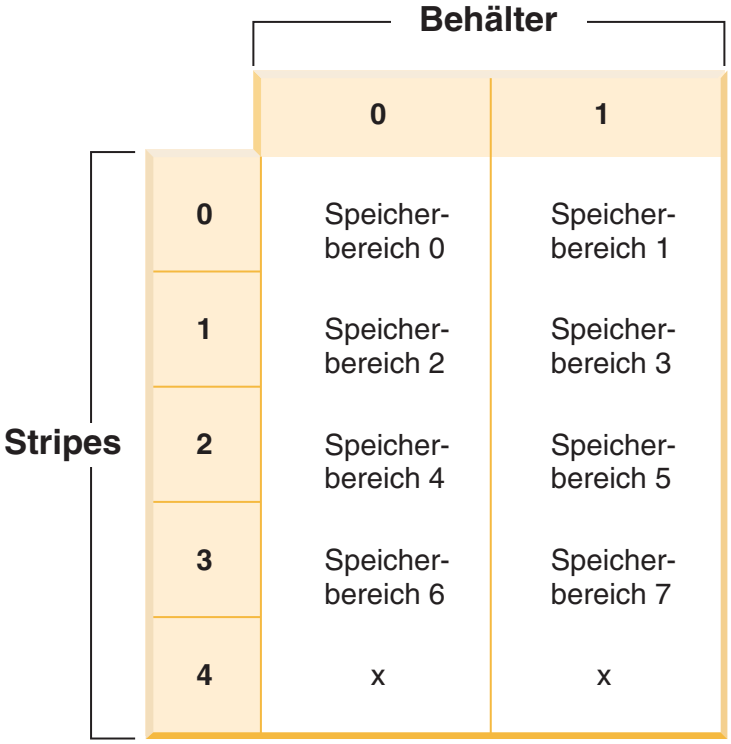


Abbildung 39. Tabellenbereich nach dem Löschen eines Behälters

Die entsprechende Tabellenbereichszuordnung, wie sie in einer Momentaufnahme des Tabellenbereichs gezeigt würde, sieht folgendermaßen aus:

```
Range     Stripe  Stripe  Max        Max    Start    End    Adj.   Containers
Number    Set     Offset  Extent     Page   Stripe   Stripe
   [0]       [0]       0       9       99        0       4    0   2 (0, 1)
```

Wenn Sie die Größe eines Behälters verringern wollen, arbeitet die Neuausgleichsfunktion in ähnlicher Weise.

Verwenden Sie zur Verkleinerung eines Behälters die Option REDUCE oder RESIZE in der Anweisung ALTER TABLESPACE. Zum Löschen eines Behälters dient die Option DROP in der Anweisung ALTER TABLESPACE.

Zugehörige Konzepte:

- „Tabellenbereichszuordnungen" auf Seite 108

Zugehörige Tasks:

- „Modifizieren von Behältern in einem DMS-Tabellenbereich" in *Systemverwaltung: Implementierung*

Zugehörige Referenzen:

- „ALTER TABLESPACE statement" in *SQL Reference, Volume 2*
- „GET SNAPSHOT Command" in *Command Reference*
- „Table space activity monitor elements" in *System Monitor Guide and Reference*

SMS- und DMS-Tabellenbereiche im Vergleich

Bei der Entscheidung, welcher Tabellenbereichstyp zum Speichern der Daten verwendet werden soll, sollten Sie das Pro und Kontra gut abwägen.

Vorteile eines SMS-Tabellenbereichs:

- Der Speicher wird vom System erst dann zugeordnet, wenn er benötigt wird.
- Beim Erstellen eines Tabellenbereichs sind weniger vorbereitende Arbeiten erforderlich, weil Sie keine Behälter vordefinieren müssen.

Vorteile eines DMS-Tabellenbereichs:

- Die Größe eines Tabellenbereichs kann durch Hinzufügen oder Erweitern von Behältern erhöht werden (mit der Anweisung ALTER TABLESPACE). Vorhandene Daten können automatisch auf die neue Gruppe von Behältern verteilt werden, um die optimale E/A-Effizienz zu erhalten.
- Eine Tabelle kann nach dem Typ der zu speichernden Daten auf mehrere Tabellenbereiche verteilt werden:
  - Langfeld- und LOB-Daten
  - Indizes
  - Reguläre Tabellendaten

  Vielleicht sollen die Tabellendaten zur Erhöhung der Leistung oder zur Vergrößerung der für eine Tabelle gespeicherten Datenmenge getrennt gehalten werden. Sie könnten beispielsweise eine Tabelle mit 64 GB für reguläre Tabellendaten, 64 GB für Indexdaten und 2 TB für Langfelddaten anlegen. Bei Verwendung von 8-KB-Seiten können die Tabellendaten und die Indexdaten bis zu 128 GB umfassen. Bei Verwendung von 16-KB-Seiten können sie bis zu 256 GB umfassen. Bei Verwendung von 32-KB-Seiten können die Tabellendaten und die Indexdaten bis zu 512 GB umfassen.
- Es ist möglich, die Speicherposition der Daten auf der Platte zu steuern, sofern das Betriebssystem dies zulässt.
- Wenn sich alle Tabellendaten in einem einzigen Tabellenbereich befinden, ist der Systemaufwand beim Löschen und erneuten Definieren eines Tabellenbereichs geringer, als dies beim Löschen und erneuten Definieren einer Tabelle der Fall wäre.
- Im Allgemeinen gilt, dass sich mit einer gut organisierten Gruppe von DMS-Tabellenbereichen eine bessere Leistung erzielen lässt als mit SMS-Tabellenbereichen.

Anmerkung: In der Solaris™-Betriebsumgebung wird die Verwendung von DMS-Tabellenbereichen mit unformatierten Einheiten (raw devices) für leistungskritische Auslastungen dringend empfohlen.

Generell lassen sich kleine Datenbanken, die von einem kleinen Personenkreis genutzt werden, am einfachsten mit SMS-Tabellenbereichen verwalten. Andererseits sollten Sie für umfangreiche, wachsende Datenbanken SMS-Tabellenbereiche nur für temporäre Tabellenbereiche und den Katalogtabellenbereich sowie getrennte DMS-Tabellenbereiche mit mehreren Behältern für jede Tabelle verwenden. Außerdem ist es wahrscheinlich sinnvoll, Langfelddaten und Indizes in eigenen Tabellenbereichen zu speichern.

Wenn Sie sich entschließen, DMS-Tabellenbereiche mit Einheitenbehältern zu verwenden, müssen Sie bereit sein, Ihre Umgebung zu optimieren und zu verwalten.

Zugehörige Konzepte:

- „Aufbau von Tabellenbereichen" auf Seite 100
- „Vom Betriebssystem verwalteter Speicherbereich (SMS)" auf Seite 104
- „Von der Datenbank verwalteter Speicherbereich (DMS)" auf Seite 106

Platten-E/A für Tabellenbereiche

Die Art und der Aufbau Ihres Tabellenbereichs bestimmen die Effizienz der Ein-/Ausgabeoperationen, die mit diesem Tabellenbereich erzielt werden kann. Es folgen einige Begriffe, mit denen Sie vertraut sein sollten, bevor Sie mit den weiteren Themen über Aufbau und Verwendung von Tabellenbereichen fortfahren:

Lesen großer Blöcke (Big Blocks)
Eine Leseoperation, bei der mehrere Seiten (in der Regel ein durch EXTENTSIZE definierter Bereich) in einer einzigen Anforderung abgerufen werden. Das Lesen mehrerer Seiten in einem Vorgang ist effizienter als das Lesen jeder Seite in getrennten Vorgängen.

Vorablesezugriff
Das Vorablesen der Seiten, auf die in einer Abfrage zugegriffen wird. Der Hauptzweck des Vorablesens ist die Verringerung von Antwortzeiten. Dies kann erreicht werden, wenn das Vorablesen von Seiten asynchron zur Ausführung der Abfrage stattfinden kann. Die besten Antwortzeiten werden erzielt, wenn entweder die CPU(s) oder das E/A-Subsystem mit maximaler Kapazität arbeiten.

Seitenlöschfunktion
Durch das Lesen und Ändern von Seiten sammeln diese sich im Pufferpool der Datenbank an. Wenn eine Datenseite eingelesen wird, wird sie in eine Seite des Pufferpools eingelesen. Wenn der Pufferpool ganz mit geänderten Seiten gefüllt ist, muss eine dieser geänderten Seiten auf die Platte geschrieben werden, bevor die neue Seite eingelesen werden kann. Bevor nun der Pufferpool gänzlich gefüllt wird, treten Seitenlöschagenten in Aktion, die geänderte Seiten auf die Platte schreiben und im Pufferpool löschen, um die Verfügbarkeit von Pufferpoolseiten für zukünftige Leseanforderungen sicherzustellen.

Immer wenn DB2® Universal Database (DB2 UDB) das Lesen großer Blöcke (Big Blocks) als vorteilhaft erkennt, werden große Blöcke gelesen. Dies tritt in der Regel beim Abrufen von Daten, die sequenzieller oder teilweise sequenzieller Art sind. Die Menge der Daten, die in einer Leseoperation gelesen werden, hängt von der Größe des Werts für EXTENTSIZE ab. Je größer der Wert für EXTENTSIZE ist, desto mehr Seiten können in einem Vorgang gelesen werden.

Die Leistung sequenzieller Vorableseoperationen kann weiterhin verbessert werden, wenn Seiten vom Datenträger in zusammenhängende Seiten im Pufferpool gelesen werden können. Da Pufferpools standardmäßig seitenbasiert sind, gibt es keine Garantie, dass sich beim Lesen vom Datenträger in zusammenhängende Seiten eine zusammenhängende Gruppe von Seiten finden lässt. Zu diesem Zweck können blockbasierte Pufferpools verwendet werden, weil diese nicht nur einen Seitenbereich, sondern außerdem einen Blockbereich für Gruppen zusammenhängender Seiten enthalten. Jede Gruppe zusammenhängender Seiten wird als Block bezeichnet, und jeder Block enthält eine Anzahl von Seiten, die als Blockgröße bezeichnet wird. Die Größen für den Seiten- und den Blockbereich sowie die Anzahl Seiten in jedem Block sind konfigurierbar.

Die Art, wie der Bereich auf der Platte gespeichert ist, hat Einfluss auf die E/A-Effizienz. In einem DMS-Tabellenbereich mit Einheitenbehältern werden die Daten eher zusammenhängend auf der Platte gespeichert und können mit minimaler Suchzeit und Plattenlatenzzeit gelesen werden. Wenn jedoch Dateien verwendet werden, können die Daten vom Dateisystem in Teile getrennt an mehr als einer Position auf der Platte gespeichert werden. Dies geschieht häufiger bei Verwendung von SMS-Tabellenbereichen, bei denen Dateien um jeweils eine Seite erweitert werden, wodurch die Fragmentierung wahrscheinlicher wird. Eine große Datei, die zur Verwendung durch einen DMS-Tabellenbereich vorab zugeordnet wurde, führt eher dazu, dass die Datei auf der Platte zusammenhängend gespeichert wird, insbesondere wenn die Datei in einem noch ungenutzten Speicherbereich zugeordnet wurde.

Sie können den Grad des Vorablesens durch Ändern des Parameters PREFETCHSIZE in der Anweisung CREATE TABLESPACE bzw. ALTER TABLESPACE steuern. (Der Standardwert für alle Tabellenbereiche in der Datenbank wird durch den Konfigurationsparameter *dft_prefetch_sz* der Datenbank festgelegt.) Der Parameter PREFETCHSIZE gibt DB2 UDB an, wie viele Seiten zu lesen sind, wenn ein Vorablesezugriff ausgelöst wird. Wenn der Wert des Parameters PREFETCHSIZE auf ein Vielfaches des Parameters EXTENTSIZE in der Anweisung CREATE TABLESPACE gesetzt wird, können mehrere EXTENTSIZE große Bereiche parallel gelesen werden. (Der Standardwert für alle Tabellenbereiche in der Datenbank wird durch den Konfigurationsparameter *dft_extent_sz* der Datenbank festgelegt.) Der Parameter EXTENTSIZE gibt die Anzahl der 4-KB-Seiten an, die in einen Behälter geschrieben werden, bevor zum nächsten Behälter übergegangen wird.

Nehmen Sie zum Beispiel an, Sie hätten einen Tabellenbereich, der drei Einheiten verwendet. Wenn Sie den Wert für PREFETCHSIZE auf das Dreifache des Werts für EXTENTSIZE setzen, kann DB2 UDB einen Lesezugriff in großen Blöcken von jeder Einheit parallel durchführen, wodurch der E/A-Durchsatz erheblich erhöht wird. Voraussetzungen sind, dass jede Einheit eine getrennte physische Einheit ist und dass der Controller über eine ausreichende Bandbreite verfügt, um den Datenstrom von jeder Einheit zu verarbeiten. Beachten Sie, dass DB2 UDB eventuell die Parameter für den Vorablesezugriff zur Laufzeit aufgrund der Abfragegeschwindigkeit, der Pufferpoolauslastung und anderer Faktoren dynamisch anpassen muss.

Einige Dateisysteme (z. B. Journaled File System unter AIX®) verfügen über eine eigene Vorablesemethode. In einigen Fällen kann der Vorablesezugriff des Dateisystems auf größere Datenmengen eingestellt sein als der Vorablesezugriff von DB2 UDB. Dies kann dazu führen, dass der Vorablesezugriff für SMS- und DMS-Tabellenbereiche mit Dateibehältern scheinbar eine bessere Leistung zeigt als der Vorablesezugriff für DMS-Tabellenbereiche mit Einheitenbehältern. Dies ist jedoch irreführend, da es sich wahrscheinlich um das Ergebnis einer zusätzlichen Ebene

des Vorablesezugriffs handelt, die innerhalb des Dateisystems wirksam ist. Normalerweise sollten DMS-Tabellenbereiche jeder äquivalenten Konfiguration im Hinblick auf die Leistung überlegen sein.

Für ein effizientes Vorablesen (oder auch nur Lesen) muss eine ausreichende Anzahl verfügbarer Pufferpoolseiten vorhanden sein. Zum Beispiel könnte es eine Anforderung zum parallelen Vorablesezugriff geben, mit dem drei (durch EXTENTSIZE bestimmte) Bereiche aus einem Tabellenbereich gelesen werden und durch den für jede einzulesende Seite eine geänderte Seite aus dem Pufferpool herausgeschrieben wird. Die Anforderung zum Vorablesezugriff könnte so weit verlangsamt werden, dass sie mit der Abfrage nicht Schritt halten kann. Daher sollten Seitenlöschfunktionen in ausreichender Anzahl konfiguriert werden, um die Anforderungen von Vorablesezugriffen erfüllen zu können.

Zugehörige Konzepte:

- „Aufbau von Tabellenbereichen" auf Seite 100
- „Vorabzugriff von Daten in den Pufferpool" in *Systemverwaltung: Optimierung*

Zugehörige Referenzen:

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

Überlegungen zur Auslastung beim Entwurf von Tabellenbereichen

Der primäre Typ der Auslastung, die von DB2® Universal Database (DB2 UDB) in Ihrer Umgebung verwaltet wird, kann Ihre Wahl beeinflussen, welcher Typ von Tabellenbereich zu verwenden und welche Seitengröße anzugeben ist. Eine OLTP-Auslastung (Online-Transaktionsverarbeitung) ist durch Transaktionen charakterisiert, die einen wahlfreien Zugriff auf Daten benötigen, häufige Einfüge- oder Aktualisierungsaktivitäten verursachen und Abfragen enthalten, die in der Regel nur kleine Datenmengen zurückliefern. In Anbetracht dessen, dass der Zugriff wahlfrei nur auf eine bzw. wenige Seiten erfolgt, ist ein Vorablesezugriff weniger wahrscheinlich.

In diesem Fall zeigen DMS-Tabellenbereiche mit Einheitenbehältern die beste Leistung. DMS-Tabellenbereiche mit Dateibehältern oder SMS-Tabellenbereiche sind ebenfalls sinnvolle Möglichkeiten für eine OLTP-Auslastung, wenn es nicht auf maximale Leistung ankommt. Wenn nur wenige oder gar keine sequenziellen E/A-Operationen zu erwarten sind, spielen die Werte der Parameter EXTENTSIZE und PREFETCHSIZE in der Anweisung CREATE TABLESPACE keine wichtige Rolle für die E/A-Effizienz. Wichtig ist jedoch, eine ausreichende Anzahl von Seitenlöschfunktionen über den Konfigurationsparameter *chngpgs_thresh* festzulegen.

Eine Abfrageauslastung wird durch Transaktionen charakterisiert, die einen sequenziellen oder teilweise sequenziellen auf Daten benötigen und in der Regel große Datenmengen zurückliefern. Ein DMS-Tabellenbereich mit mehreren Einheitenbehältern (wobei sich jeder Behälter auf einer separaten Platte befindet) bietet das größte Potenzial für einen effizienten parallelen Vorablesezugriff. Der Wert des Parameters PREFETCHSIZE in der Anweisung CREATE TABLESPACE sollte das Produkt aus dem Wert des Parameters EXTENTSIZE multipliziert mit der Anzahl der Einheitenbehälter sein. Dadurch kann DB2 UDB aus allen Behältern parallel vorablesen. Wenn sich die Anzahl von Behältern ändert oder der Vorab-

lesezugriff aus einem Grund verstärkt oder verringert werden muss, kann der Wert für PREFETCHSIZE über die Anweisung ALTER TABLESPACE entsprechend geändert werden.

Eine sinnvolle Alternative für eine Abfrageauslastung ist die Verwendung von Dateien, wenn das Dateisystem über eine eigene Vorablesefunktion verfügt. Die Dateien können entweder zu DMS-Tabellenbereichen mit Dateibehältern gehören oder Dateien für SMS-Tabellenbereiche sein. Beachten Sie, dass Sie bei Verwendung von SMS sicherstellen müssen, dass die Verzeichnisbehälter getrennten physischen Datenträgern zugeordnet sind, um E/A-Parallelität zu erreichen.

Das Ziel für eine gemischte Auslastung besteht darin, einzelne E/A-Anforderungen so effizient wie möglich für OLTP-Auslastungen zu machen und die Effizienz paralleler E/A-Operationen für Abfrageauslastungen zu maximieren.

Beim Ermitteln der Seitengröße für einen Tabellenbereich sind folgende Überlegungen zu berücksichtigen:

- Für OLTP-Anwendungen, die wahlfreie Lese- und Schreiboperationen durchführen, ist eine geringere Seitengröße empfehlenswert, weil dabei weniger Pufferspeicher durch unerwünschte Zeilen belegt wird.
- Für DSS-Anwendungen (Decision Support System), die jeweils auf eine große Anzahl aufeinander folgender Zeilen gleichzeitig zugreifen, sind größere Seiten empfehlenswert, weil dadurch weniger E/A-Anforderungen erforderlich sind, um eine bestimmte Anzahl Zeilen zu lesen. Es gibt jedoch eine Ausnahme von dieser Regel. Diese betrifft den Fall, dass die Zeilengröße kleiner als folgender Wert ist:

  ```
  seitengröße / 255
  ```

  Bei dieser Größe bleibt auf jeder Seite Speicherbereich ungenutzt (weil jede Seite maximal 255 Zeilen enthalten kann). In diesem Fall ist eine geringere Seitengröße zu empfehlen.
- Durch größere Seiten können Sie möglicherweise die Zahl der Indexstufen reduzieren.
- Größere Seiten unterstützen längere Zeilen.
- Auf 4-KB-Standardseiten sind Tabellen auf 500 Spalten begrenzt, während auf größeren Seiten (8 KB, 16 KB und 32 KB) 1012 Spalten unterstützt werden.
- Die Maximalgröße des Tabellenbereichs ist proportional zur Seitengröße des Tabellenbereichs.

Zugehörige Konzepte:

- „Vom Betriebssystem verwalteter Speicherbereich (SMS)" auf Seite 104
- „Von der Datenbank verwalteter Speicherbereich (DMS)" auf Seite 106

Zugehörige Referenzen:

- „chngpgs_thresh - Schwellenwert für geänderte Seiten" in *Systemverwaltung: Optimierung*
- „ALTER TABLESPACE statement" in *SQL Reference, Volume 2*
- „CREATE TABLESPACE statement" in *SQL Reference, Volume 2*
- „SQL limits" in *SQL Reference, Volume 1*

Parameter EXTENTSIZE

Der Wert des Parameters EXTENTSIZE für einen Tabellenbereich gibt die Anzahl der Seiten mit Tabellendaten an, die in einen Behälter geschrieben werden, bevor Daten in den nächsten Behälter geschrieben werden. Beim Auswählen eines Werts für EXTENTSIZE ist Folgendes zu beachten:

- Größe und Typ der Tabellen im Tabellenbereich

  In DMS-Tabellenbereichen wird einer Tabelle gleichzeitig jeweils ein durch EXTENTSIZE definierter Speicherbereich zugeordnet. Wenn beim Füllen der Tabelle mit Daten ein EXTENTSIZE großer Bereich voll ist, wird ein neuer Bereich dieser Größe zugeordnet. Speicher in DMS-Tabellenbereichsbehältern wird vorab reserviert. Das bedeutet, dass neue EXTENTSIZE große Speicherbereiche zugeordnet werden, bis der Behälter vollständig gefüllt ist.

  Speicherbereich in SMS-Tabellenbereichen wird einer Tabelle jeweils in der Größe eines EXTENTSIZE-Bereichs oder einer Seite zugeordnet. Wenn beim Füllen der Tabelle mit Daten ein EXTENTSIZE großer Bereich oder eine Seite voll ist, wird ein neuer Bereich oder eine neue Seite zugeordnet, bis alle EXTENTSIZE-Bereiche oder Seiten in dem betreffenden Dateisystem belegt sind. Bei der Verwendung von SMS-Tabellenbereichen ist die mehrseitige Dateizuordnung zulässig. Die mehrseitige Dateizuordnung ermöglicht die Zuordnung jeweils EXTENTSIZE großer Speicherbereiche anstelle einzelner Seiten.

  Anmerkung: Die mehrseitige Dateizuordnung wird für alle SMS-Tabellenbereiche in einer Datenbank über das Dienstprogramm `db2empfa` aktiviert. Wenn die mehrseitige Dateizuordnung aktiviert ist, kann sie nicht mehr inaktiviert werden. Vor Version 8.2 wurden Datenbanken standardmäßig mit inaktivierter mehrseitiger Dateizuordnung erstellt. In Version 8.2 wurde diese Standardeinstellung geändert, so dass Datenbanken standardmäßig mit aktivierter mehrseitiger Dateizuordnung erstellt werden, da dies zu Leistungsverbesserungen führt. Wenn Sie die Verhaltensweise der Versionen vor Version 8.2 aktivieren möchten, setzen Sie die Registriervariable DB2_NO_MPFA_FOR_NEW_DB auf den Wert ON.

  Eine Tabelle besteht aus folgenden separaten Tabellenobjekten:

  – Ein Datenobjekt. Hier werden die regulären Spaltendaten gespeichert.
  – Ein Indexobjekt. Hier werden alle für die Tabelle definierten Indizes gespeichert.
  – Ein Langfeldobjekt. Hier werden Langfelddaten gespeichert, wenn die Tabelle eine oder mehrere Spalten mit LONG-Datentypen besitzt.
  – Zwei LOB-Objekte. Wenn die Tabelle eine oder mehrere LOB-Spalten hat, werden diese in folgenden beiden Tabellenobjekten gespeichert:
    - Ein Tabellenobjekt für die LOB-Daten
    - Ein zweites Tabellenobjekt für Metadaten, die die LOB-Daten beschreiben
  – Ein Blockzuordnungsobjekt für mehrdimensionale Tabellen

  Jedes Tabellenobjekt wird getrennt gespeichert und ordnet nach Bedarf neue (durch EXTENTSIZE definierte) Bereiche zu. Jedes DMS-Tabellenobjekt wird außerdem mit einem Metadatenobjekt verbunden, das als *Speicherbereichsmaske* bezeichnet wird und alle EXTENTSIZE großen Bereiche im Tabellenbereich beschreibt, die zu dem Tabellenobjekt gehören. Der Speicherbereich für Speicherbereichsmasken wird ebenfalls jeweils in der Größe von EXTENTSIZE zugeordnet.

Daher beträgt die Anfangszuordnung von Speicherbereich für ein Objekt in einem DMS-Tabellenbereich zwei EXTENTSIZE-Größen. (Die Anfangszuordnung von Speicherbereich für ein Objekt in einem SMS-Tabellenbereich beträgt eine Seite). Wenn Sie also viele kleine Tabellen in einem DMS-Tabellenbereich haben, ist es möglich, dass eine relativ große Menge an Speicher für eine relativ kleine Menge von Daten zugeordnet ist. In einem solchen Fall sollten Sie einen kleinen Wert für EXTENTSIZE definieren.

Wenn Sie andererseits eine sehr umfangreiche Tabelle mit einer hohen Zuwachsrate haben und einen DMS-Tabellenbereich mit einem niedrigen Wert für EXTENTSIZE verwenden, könnte durch häufiges Zuordnen zusätzlicher Speicherbereiche unnötiger Systemaufwand entstehen.

- Art des Zugriffs auf die Tabellen

  Wenn auf die Tabellen durch zahlreiche Abfragen oder Transaktionen zugegriffen wird, die große Datenmengen verarbeiten, kann das Vorablesen von Daten aus den Tabellen eine wesentliche Leistungsverbesserung bewirken.
- Minimal erforderliche Anzahl von EXTENTSIZE-Speicherbereichen

  Falls nicht genügend Platz für fünf EXTENTSIZE-Speicherbereiche des Tabellenbereichs in den Behältern vorhanden ist, wird der Tabellenbereich nicht erstellt.

Zugehörige Konzepte:

- „Aufbau von Tabellenbereichen" auf Seite 100

Zugehörige Referenzen:

- „CREATE TABLESPACE statement" in *SQL Reference, Volume 2*
- „db2empfa - Enable Multipage File Allocation Command" in *Command Reference*

Beziehung zwischen Tabellenbereichen und Pufferpools

Jeder Tabellenbereich wird einem bestimmten Pufferpool zugeordnet. Der Standardpufferpool ist IBMDEFAULTBP. Wenn ein anderer Pufferpool einem Tabellenbereich zugeordnet werden soll, muss der Pufferpool existieren (er wird mit der Anweisung CREATE BUFFERPOOL definiert) und die gleiche Seitengröße besitzen. Die Zuordnung wird bei der Erstellung des Tabellenbereichs (mit der Anweisung CREATE TABLESPACE) definiert. Die Zuordnung zwischen dem Tabellenbereich und dem Pufferpool kann mit der Anweisung ALTER TABLESPACE geändert werden.

Durch die Verwendung mehrerer Pufferpools haben Sie die Möglichkeit, die Verwendung von Hauptspeicher durch die Datenbank zu konfigurieren, um die allgemeine Leistung zu verbessern. Zum Beispiel kann bei Tabellenbereichen mit einer oder mehreren umfangreichen Tabellen (d. h. größeren als der verfügbare Hauptspeicher), auf die die Benutzer wahlfrei zugreifen, die Größe des Pufferpools begrenzt werden, da ein Zwischenspeichern (Caching) der Datenseiten vielleicht nicht vorteilhaft ist. Dem Tabellenbereich für eine Online-Transaktionsanwendung kann ein größerer Pufferpool zugeordnet werden, so dass die Datenseiten, die von der Anwendung genutzt werden, länger im Cache zwischengespeichert und so schnellere Antwortzeiten erzielt werden können. Vorsicht ist jedoch bei der Konfiguration neuer Pufferpools geboten.

Anmerkung: Wenn Sie ermittelt haben, dass für Ihre Datenbank eine Seitengröße von 8 KB, 16 KB oder 32 KB erforderlich ist, muss jeder Tabellenbereich mit einer dieser Seitengrößen einem Pufferpool mit derselben Seitengröße zugeordnet werden.

Der Speicher, der für alle Pufferpools benötigt wird, muss dem Datenbankmanager bereits beim Starten der Datenbank zur Verfügung stehen. Wenn DB2® Universal Database (DB2 UDB) den erforderlichen Speicherbereich nicht erhalten kann, startet der Datenbankmanager mit den Standardpufferpools (je einem mit 4-KB-Seiten, 8-KB-Seiten, 16-KB-Seiten und 32-KB-Seiten) und gibt eine Warnung aus.

In einer Umgebung mit partitionierten Datenbanken können Sie einen Pufferpool mit derselben Größe für alle Partitionen in der Datenbank erstellen. Sie können auch Pufferpools verschiedener Größen in verschiedenen Partitionen erstellen.

Zugehörige Konzepte:

- „Table spaces and other storage structures" in *SQL Reference, Volume 1*

Zugehörige Referenzen:

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

Beziehung zwischen Tabellenbereichen und Datenbankpartitionsgruppen

In einer Umgebung mit partitionierten Datenbanken wird jeder Tabellenbereich einer bestimmten Datenbankpartitionsgruppe zugeordnet. Dadurch können die Merkmale des Tabellenbereichs auf jede Partition in der Datenbankpartitionsgruppe angewendet werden. Die Datenbankpartitionsgruppe muss vorhanden sein (sie wird mit der Anweisung CREATE DATABASE PARTITION GROUP definiert). Die Zuordnung zwischen dem Tabellenbereich und der Datenbankpartitionsgruppe wird bei der Erstellung des Tabellenbereichs über die Anweisung CREATE TABLESPACE definiert.

Die Zuordnung zwischen dem Tabellenbereich und der Datenbankpartitionsgruppe kann mit der Anweisung ALTER TABLESPACE nicht geändert werden. Sie können lediglich die Spezifikation des Tabellenbereichs für einzelne Partitionen innerhalb der Datenbankpartitionsgruppe ändern. In einer Umgebung mit einer Einzelpartition wird jeder Tabellenbereich der Standarddatenbankpartitionsgruppe zugeordnet. Die Standarddatenbankpartitionsgruppe bei der Definition eines Tabellenbereichs ist IBMDEFAULTGROUP, sofern kein temporärer Systemtabellenbereich definiert wird. In diesem Fall wird die Standarddatenbankpartitionsgruppe IBMTEMPGROUP verwendet.

Zugehörige Konzepte:

- „Table spaces and other storage structures" in *SQL Reference, Volume 1*
- „Datenbankpartitionsgruppen" auf Seite 91
- „Aufbau von Tabellenbereichen" auf Seite 100

Zugehörige Referenzen:

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

Speicherverwaltungssicht

Verwenden Sie die Speicherverwaltungssicht zur Überwachung des Speicherstatus einer partitionierten Datenbank. Diese Sicht ist die grafische Oberfläche für das Speicherverwaltungstool. In der Speicherverwaltungssicht können Sie Speichermomentaufnahmen einer Datenbank, einer Datenbankpartitionsgruppe oder eines Tabellenbereichs aufzeichnen. Beim Erstellen einer Momentaufnahme eines Tabellenbereichs werden Statistikdaten aus dem Systemkatalog und dem Datenbankmonitor für Tabellen, Indizes und Behälter erfasst, die für den angegebenen Tabellenbereich definiert wurden. Bei einer Momentaufnahme einer Datenbank oder Datenbankpartitionsgruppe werden Statistikdaten für alle Tabellenbereiche erfasst, die in der vorliegenden Datenbank oder Datenbankpartitionsgruppe definiert sind. Bei der Momentaufnahme einer Datenbank werden Statistikdaten für alle Datenbankpartitionsgruppen in der Datenbank erfasst. Mit den unterschiedlichen Speichermomentaufnahmen können unterschiedliche Speicheraspekte überwacht werden:

- Die Speichernutzung kann am besten mit Momentaufnahmen von Tabellenbereichen überwacht werden.
- Nur bei partitionierten Datenbanken: Die Datenabweichung (Datenbankverteilung) kann am besten mit Momentaufnahmen von Datenbankpartitionsgruppen überwacht werden.
- Das Clusterverhältnis von Indizes kann sowohl mit Momentaufnahmen von Datenbankpartitionsgruppen als auch von Tabellenbereichen erfasst werden. Das Clusterverhältnis von Indizes wird in der Detailsicht des Indexordners angezeigt.

In der Speicherverwaltungssicht können Sie außerdem Schwellenwerte für die Datenabweichung, die Speichernutzung und das Indexclusterverhältnis angeben. Wenn ein Zielobjekt einen angegebenen Schwellenwert überschreitet, werden das neben dem betreffenden Objekt dargestellte Symbol und das Symbol seines übergeordneten Objekts in der Speicherverwaltungssicht mit einer Warnungsmarkierung oder einer Alarmmarkierung gekennzeichnet.

Anmerkung: Sie können nur Schwellenwerte für Datenabweichungen für partitionierte Datenbanken festlegen.

Zugehörige Referenzen:

- „Gespeicherte Prozeduren für das Speicherverwaltungstool" auf Seite 133
- „Sichttabellen zur Speicherverwaltung" auf Seite 134

Gespeicherte Prozeduren für das Speicherverwaltungstool

Die folgende Tabelle zeigt die Funktionen der gespeicherten Prozeduren, die für das Speicherverwaltungstool erstellt werden. Die gespeicherten Prozeduren werden bei der Erstellung der Datenbank automatisch erstellt. Außerdem werden ihre jeweiligen Pakete bei Bedarf gebunden.

Tabelle 21. Gespeicherte Prozeduren für das Speicherverwaltungstool

Vollständig qualifizierter Name	Parameter	Funktionalität
SYSPROC.CREATE_STORAGEMGMT _TABLES	in_tbspace VARCHAR(128) Eingabe - Tabellenbereichs-name	Erstellt alle Speicher-verwaltungstabellen unter dem festen Schema ″DB2TOOLS″ in dem durch die Eingabe angegebenen Tabellenbereich.
SYSPROC.DROP_STORAGEMGMT _TABLES	dropSpec SMALLINT Eingabe - 0 / 1	Versucht, alle Speicher-verwaltungstabellen zu löschen. Wenn dropSpec=0, wird der Prozess gestoppt, falls irgendein Fehler auftritt. Wenn dropSpec=1, wird der Prozess fortgesetzt und das Auftreten aller Fehler igno-riert.
SYSPROC.CAPTURE_STORAGEMGMT _INFO	in_rootType SMALLINT Eingabe - alle gültigen Werte sind in der Tabelle STMG_OBJECT_TYPE angegeben in_rootSchema VARCHAR(128) Eingabe - Schemaname des Stamm-objekts der Speicher-momentaufnahme in_rootName VARCHAR(128) Eingabe - Name des Stamm-objekts	Versucht, für den System-katalog und die Momentauf-nahme speicherspezifische Informationen für das gege-bene Stammobjekt sowie für die in seinem Bereich definier-ten Speicherobjekte zu erfas-sen. Alle Speicherobjekte sind in der Tabelle STMG_OBJECT_TYPE angegeben.

Zugehörige Referenzen:

- „Speicherverwaltungssicht″ auf Seite 133

Sichttabellen zur Speicherverwaltung

Tabelle STMG_OBJECT_TYPE:

Die Tabelle STMG_OBJECT_TYPE enthält eine Zeile für jeden unterstützten Speichertyp, der überwacht werden kann.

Tabelle 22. Tabelle STMG_OBJECT_TYPE

Spaltenname	Datentyp	Nullwert?	Beschreibung
OBJ_TYPE	INTEGER	N	Integerwert entspricht einem Typ von Speicherobjekt.
TYPE_NAME	VARCHAR	N	Beschreibender Name des Speicher-objekttyps

Tabelle STMG_THRESHOLD_REGISTRY:

Die Tabelle STMG_THRESHOLD_REGISTRY enthält eine Zeile für jeden Speicherschwellenwerttyp. Die aktivierten Schwellenwerte werden bei der Erstellung einer Speichermomentaufnahme durch den Analyseprozess verwendet.

Tabelle 23. Tabelle STMG_THRESHOLD_REGISTRY

Spaltenname	Datentyp	Nullwert?	Beschreibung
STMG_TH_TYPE	INTEGER	N	Integerwert entspricht einem Speicherschwellenwerttyp.
ENABLED	CHARACTER	N	Y = Schwellenwert ist aktiviert N = Schwellenwert ist nicht aktiviert und wird bei der Speicheranalyse nicht zum Vergleich herangezogen
STMG_TH_NAME	VARCHAR	J	Beschreibender Name des Speicherschwellenwerts

Tabelle STMG_CURR_THRESHOLD:

Die Tabelle STMG_CURR_THRESHOLD enthält eine Zeile für jeden Schwellenwerttyp, der explizit für ein Speicherobjekt festgelegt ist.

Tabelle 24. Tabelle STMG_CURR_THRESHOLD

Spaltenname	Datentyp	Nullwert?	Beschreibung
STMG_TH_TYPE	INTEGER	N	Integerwert entspricht einem Speicherschwellenwerttyp.
OBJ_TYPE	INTEGER	N	Integerwert entspricht einem Typ von Speicherobjekt.
OBJ_NAME	VARCHAR	N	Der Name des Speicherobjekts
OBJ_SCHEMA	VARCHAR	N	Das Schema des Speicherobjekts. ″-″ wird verwendet, wenn ein Schema für das Objekt nicht zutreffend ist
WARNING_THRESHOLD	SMALLINT	J	Der Wert des Warnungsschwellenwerts, der für das Speicherobjekt definiert ist
ALARM_THRESHOLD	SMALLINT	J	Der Wert des Alarmschwellenwerts, der für das Speicherobjekt definiert ist

Tabelle STMG_ROOT_OBJECT:

Die Tabelle STMG_ROOT_OBJECT enthält eine Zeile für das Stammobjekt jeder Speichermomentaufnahme.

Tabelle 25. Tabelle STMG_ROOT_OBJECT

Spaltenname	Datentyp	Nullwert?	Beschreibung
STMG_TIMESTAMP	TIMESTAMP	N	Die Zeitmarke der Speichermomentaufnahme. Sie gibt an, wann der Datenerfassungsprozess gestartet wurde.
OBJ_TYPE	INTEGER	N	Integerwert entspricht einem Typ von Speicherobjekt.
ROOT_ID	VARCHAR	N	Die ID des Stammobjekts

Tabelle STMG_OBJECT:

Die Tabelle STMG_OBJECT enthält eine Zeile für jedes Speicherobjekt, das durch die bisher erfassten Speichermomentaufnahmen analysiert wurde.

Tabelle 26. Tabelle STMG_OBJECT

Spaltenname	Datentyp	Nullwert?	Beschreibung
STMG_TIMESTAMP	TIMESTAMP	N	Die Zeitmarke der Speicher-momentaufnahme. Sie gibt den Zeitpunkt an, zu dem der Datenerfassungsprozess gestartet wurde.
ROOT_ID	CHARACTER	N	Die ID des Stammobjekts
OBJ_TYPE	INTEGER	N	Integerwert entspricht einem Typ von Speicherobjekt.
OBJ_SCHEMA	VARCHAR	N	Das Schema des Speicherobjekts. "-" wird verwendet, wenn ein Schema für das Objekt nicht zutreffend ist
OBJ_NAME	VARCHAR	N	Der Name des Speicherobjekts
DBPG_NAME	VARCHAR	J	Der Name der Datenbankpartitions-gruppe, in der sich das Objekt befin-det. Null, falls nicht zutreffend.
TS_NAME	VARCHAR	J	Der Name des Tabellenbereichs, in dem sich das Objekt befindet. Null, falls nicht zutreffend.
OBJ_ID	VARCHAR	N	Die eindeutige Kennung für jedes Speicherobjekt unter einer gegebenen Zeitmarke einer Speicher-momentaufnahme

Tabelle STMG_HIST_THRESHOLD:

Die Tabelle STMG_HIST_THRESHOLD enthält eine Zeile für jeden Schwellenwert, der zur Analyse der Speicherobjekte zum Zeitpunkt der Erfassung der Speicher-momentaufnahmen verwendet wird.

Tabelle 27. Tabelle STMG_HIST_THRESHOLD

Spaltenname	Datentyp	Nullwert?	Beschreibung
STMG_TIMESTAMP	TIMESTAMP	N	Die Zeitmarke der Speicher-momentaufnahme. Sie gibt den Zeitpunkt an, zu dem der Datenerfassungsprozess gestartet wurde.
STMG_TH_TYPE	INTEGER	N	Integerwert entspricht einem Speicher-schwellenwerttyp.
OBJ_ID	VARCHAR	N	Die eindeutige Kennung für jedes Speicherobjekt unter einer gegebenen Zeitmarke einer Speicher-momentaufnahme

Tabelle 27. Tabelle STMG_HIST_THRESHOLD (Forts.)

Spaltenname	Datentyp	Nullwert?	Beschreibung
WARNING_THRESHOLD	SMALLINT	J	Der Wert des Warnungsschwellenwerts, der für das Speicherobjekt zum Zeitpunkt der Erfassung der Speichermomentaufnahme festgelegt war
ALARM_THRESHOLD	SMALLINT	J	Der Wert des Alarmschwellenwerts, der für das Speicherobjekt zum Zeitpunkt der Erfassung der Speichermomentaufnahme festgelegt war

Tabelle STMG_DATABASE:

Die Tabelle STMG_DATABASE enthält eine Zeile für jeden detaillierten Eintrag von Datenbankspeichermomentaufnahmen.

Tabelle 28. Tabelle STMG_DATABASE

Spaltenname	Datentyp	Nullwert?	Beschreibung
STMG_TIMESTAMP	TIMESTAMP	N	Die Zeitmarke der Speichermomentaufnahme. Sie gibt an, wann der Datenerfassungsprozess gestartet wurde.
OBJ_ID	VARCHAR	N	Die eindeutige Kennung für jedes Speicherobjekt unter einer gegebenen Zeitmarke einer Speichermomentaufnahme
COMPLETE_TIMESTAMP	TIMESTAMP	J	Die Zeitmarke für den Zeitpunkt, zu dem der in der Spalte OBJ_ID bezeichnete Datenerfassungsprozess für die Datenbank abgeschlossen wurde
SPACE_THRESHOLD_EXCEEDED	SMALLINT	J	Eine Markierung, die den Status der Speicherbelegung der Datenbank angibt 0 = normaler Betriebsstatus; 1 = Warnungsschwellenwert überschritten; 2 = Alarmschwellenwert überschritten
SKEW_THRESHOLD_EXCEEDED	SMALLINT	J	Eine Markierung, die den Datenverteilungsstatus der Datenbank angibt 0 = normaler Betriebsstatus; 1 = Warnungsschwellenwert überschritten; 2 = Alarmschwellenwert überschritten
CR_THRESHOLD_EXCEEDED	SMALLINT	J	Eine Markierung, die den Index-Clusterbildungsstatus der Datenbank angibt 0 = normaler Betriebsstatus; 1 = Warnungsschwellenwert überschritten; 2 = Alarmschwellenwert überschritten

Tabelle STMG_DBPGROUP:

Die Tabelle STMG_DBPGROUP enthält eine Zeile für jeden detaillierten Eintrag von Speichermomentaufnahmen für Datenbankpartitionsgruppen.

Tabelle 29. Tabelle STMG_DBPGROUP

Spaltenname	Datentyp	Nullwert?	Beschreibung
STMG_TIMESTAMP	TIMESTAMP	N	Die Zeitmarke der Speichermomentaufnahme. Sie gibt an, wann der Datenerfassungsprozess gestartet wurde.
OBJ_ID	VARCHAR	N	Die eindeutige Kennung für jedes Speicherobjekt unter einer gegebenen Zeitmarke einer Speichermomentaufnahme
COMPLETE_TIMESTAMP	TIMESTAMP	J	Die Zeitmarke für den Zeitpunkt, zu dem der in der Spalte OBJ_ID bezeichnete Datenerfassungsprozess für die Datenbankpartitionsgruppe abgeschlossen wurde
SPACE_THRESHOLD_EXCEEDED	SMALLINT	J	Eine Markierung, die den Status der Speicherbelegung der Datenbankpartitionsgruppe angibt 0 = normaler Betriebsstatus; 1 = Warnungsschwellenwert überschritten; 2 = Alarmschwellenwert überschritten
SKEW_THRESHOLD_EXCEEDED	SMALLINT	J	Eine Markierung, die den Datenverteilungsstatus der Datenbankpartitionsgruppe angibt 0 = normaler Betriebsstatus; 1 = Warnungsschwellenwert überschritten; 2 = Alarmschwellenwert überschritten
PARTITON_COUNT	SMALLINT	J	Die Anzahl der in der Datenbankpartitionsgruppe enthaltenen Partitionen
TARGET_LEVEL	BIGINT	J	Die durchschnittliche Datengröße (in Byte) für alle Partitionen, die in der Datenbankpartitionsgruppe enthalten sind. Dies ist die Zieldatengröße einer gleichmäßigen Datenverteilung.
DATA_SKEW	SMALLINT	J	Ein Prozentwert für die maximale Datengrößenabweichung vom TARGET_LEVEL-Wert unter allen Partitionen. Dieser Wert wird beim Datenerfassungs- und Analyseprozess zum Vergleich mit dem Wert für die Abweichung der Datenverteilung verwendet, der für die Datenbankpartitionsgruppe in der Tabelle STMG_CURR_THRESHOLD festgelegt ist.

Tabelle 29. Tabelle STMG_DBPGROUP (Forts.)

Spaltenname	Datentyp	Nullwert?	Beschreibung
TOTAL_SIZE	BIGINT	J	Die Gesamtgröße (in Byte) für alle Partitionen, die in der Datenbankpartitionsgruppe enthalten sind. Dies ist die Summe der Gesamtgrößen (d. h. jeweils die Anzahl der Seiten multipliziert mit der Seitengröße) aller Tabellenbereiche, die unter der Datenbankpartitionsgruppe definiert sind. Für DMS-Tabellenbereiche ist die Gesamtgröße die zugeordnete Größe; für SMS-Tabellenbereiche ist die Gesamtgröße die zurzeit vom Tabellenbereich belegte Größe.
DATA_SIZE	BIGINT	J	Die Datengröße (in Byte) für alle Partitionen, die in der Datenbankpartitionsgruppe enthalten sind. Dies ist die Summe der Datengrößen (d. h. jeweils die Anzahl der Datenseiten multipliziert mit der Seitengröße) aller Tabellenbereiche, die unter der Datenbankpartitionsgruppe definiert sind.
PERCENT_USED	SMALLINT	J	Ein Prozentwert der Datengröße im Verhältnis zur Gesamtgröße. Dieser Wert wird beim Datenerfassungs- und Analyseprozess mit dem Schwellenwert für die Speichernutzung verglichen. Bei SMS-Tabellenbereichen sollte der Schwellenwert für die Speichernutzung für den Tabellenbereich oder seine übergeordnete Datenbankpartitionsgruppe auf 100 gesetzt werden, um unnötige Alarmnachrichten zu vermeiden.

Tabelle STMG_DBPARTITION:

Die Tabelle STMG_DBPARTITION enthält eine Zeile für jeden detaillierten Eintrag von Momentaufnahmen von Datenbankpartitionsgruppen. Diese Tabelle ist zur Verwendung zusammen mit der Tabelle STMG_DBPGROUP vorgesehen.

Tabelle 30. Tabelle STMG_DBPARTITION

Spaltenname	Datentyp	Nullwert?	Beschreibung
STMG_TIMESTAMP	TIMESTAMP	N	Die Zeitmarke der Speichermomentaufnahme. Sie gibt an, wann der Datenerfassungsprozess gestartet wurde.
OBJ_ID	VARCHAR	N	Die eindeutige Kennung für jedes Speicherobjekt unter einer gegebenen Zeitmarke einer Speichermomentaufnahme
PARTITION_NUM	INTEGER	J	Die Datenbankpartitionsnummer

Tabelle 30. Tabelle STMG_DBPARTITION (Forts.)

Spaltenname	Datentyp	Nullwert?	Beschreibung
COMPLETE_TIMESTAMP	TIMESTAMP	J	Die Zeitmarke für den Zeitpunkt, zu dem der in der Spalte OBJ_ID bezeichnete Datenerfassungsprozess für die Datenbankpartition abgeschlossen wurde
DBPG_NAME	CHARACTER	J	Der Name der Datenbankpartitionsgruppe
IN_USE	CHARACTER	J	Status der Partition zum Zeitpunkt der Erfassung der Speichermomentaufnahme. Stimmt mit der Spalte IN_USE in SYSCAT.NODEGROUPDEF überein.
HOST_NAME	VARCHAR	J	Der Hostname der Datenbankpartition
HOST_SYSTEM_SIZE	BIGINT	J	NICHT VERFÜGBAR
EST_DATA_SIZE	BIGINT	J	Die geschätzte Datengröße in der Datenbankpartition innerhalb des Bereichs der Datenbankpartitionsgruppe. Dieser Wert wird als Summe der Datengrößen der Tabellenpartitionen für die jeweilige Partition berechnet.

Tabelle STMG_TABLESPACE:

Die Tabelle STMG_TABLESPACE enthält eine Zeile für jeden detaillierten Eintrag von Momentaufnahmen für Tabellenbereiche.

Tabelle 31. Tabelle STMG_TABLESPACE

Spaltenname	Datentyp	Nullwert?	Beschreibung
STMG_TIMESTAMP	TIMESTAMP	N	Die Zeitmarke der Speichermomentaufnahme. Sie gibt an, wann der Datenerfassungsprozess gestartet wurde.
OBJ_ID	VARCHAR	N	Die eindeutige Kennung für jedes Speicherobjekt unter einer gegebenen Zeitmarke einer Speichermomentaufnahme
COMPLETE_TIMESTAMP	TIMESTAMP	J	Die Zeitmarke für den Zeitpunkt, zu dem der in der Spalte OBJ_ID bezeichnete Datenerfassungsprozess für den Tabellenbereich abgeschlossen wurde
SPACE_THRESHOLD_EXCEEDED	SMALLINT	J	Eine Markierung, die den Status der Speicherbelegung des Tabellenbereichs angibt 0 = normaler Betriebsstatus; 1 = Warnungsschwellenwert überschritten; 2 = Alarmschwellenwert überschritten
TYPE	CHARACTER	J	Wie in SYSCAT.TABLESPACES definiert
DATATYPE	CHARACTER	J	Wie in SYSCAT.TABLESPACES definiert

Tabelle 31. Tabelle STMG_TABLESPACE (Forts.)

Spaltenname	Datentyp	Nullwert?	Beschreibung
TOTAL_SIZE	BIGINT	J	Wie in SYSCAT.TABLESPACES definiert
PERCENT_USED	SMALLINT	J	Wie in SYSCAT.TABLESPACES definiert. Dieser Wert wird beim Datenerfassungs- und Analyseprozess zum Vergleich mit dem Schwellenwert für die Speicherbelegung in der Tabelle STMG_CURR_THRESHOLD verwendet.
DATA_SIZE	BIGINT	J	Wie in SYSCAT.TABLESPACES definiert
DATA_PAGE	BIGINT	J	Wie in SYSCAT.TABLESPACES definiert
EXTENT_SIZE	INTEGER	J	Wie in SYSCAT.TABLESPACES definiert
PREFETCH_SIZE	INTEGER	J	Wie in SYSCAT.TABLESPACES definiert
OVERHEAD	DOUBLE	J	Wie in SYSCAT.TABLESPACES definiert
TRANSFER_RATE	DOUBLE	J	Wie in SYSCAT.TABLESPACES definiert
BUFFERPOOL_ID	INTEGER	J	Wie in SYSCAT.TABLESPACES definiert
PAGE_SIZE	INTEGER	J	Wie in SYSCAT.TABLESPACES definiert

Tabelle STMG_CONTAINER:

Die Tabelle STMG_CONTAINER enthält eine Zeile für jeden detaillierten Eintrag von Momentaufnahmen für Behälter.

Tabelle 32. Tabelle STMG_CONTAINER

Spaltenname	Datentyp	Nullwert?	Beschreibung
STMG_TIMESTAMP	TIMESTAMP	N	Die Zeitmarke der Speichermomentaufnahme. Sie gibt an, wann der Datenerfassungsprozess gestartet wurde.
OBJ_ID	VARCHAR	N	Die eindeutige Kennung für jedes Speicherobjekt unter einer gegebenen Zeitmarke einer Speichermomentaufnahme
COMPLETE_TIMESTAMP	TIMESTAMP	J	Die Zeitmarke für den Zeitpunkt, zu dem der in der Spalte OBJ_ID bezeichnete Datenerfassungsprozess für den Behälter abgeschlossen wurde
TS_ID	INTEGER	J	Die Integer-ID für den Tabellenbereich, dem der Behälter zugeordnet ist

Tabelle 32. Tabelle STMG_CONTAINER (Forts.)

Spaltenname	Datentyp	Nullwert?	Beschreibung
SPACE_THRESHOLD_EXCEEDED	SMALLINT	J	Eine Markierung, die den Status der Speicherbelegung des Behälters angibt 0 = normaler Betriebsstatus; 1 = Warnungsschwellenwert überschritten; 2 = Alarmschwellenwert überschritten
PARTITION_NUM	INTEGER	J	Die Partitionsnummer der Datenbankpartition, in der sich der Behälter befindet
TYPE	CHARACTER	J	'P' = Pfadbehälter; 'F' = Dateibehälter; 'D' = unformatierter Einheitenbehälter
TOTAL_PAGE	BIGINT	J	Gesamtzahl der dem Behälter zugeordneten Seiten
USABLE_PAGES	BIGINT	J	Anzahl der verwendbaren Seiten im Behälter
PERCENT_USED	SMALLINT	J	NICHT VERFÜGBAR. Dieser Wert soll beim Datenerfassungs- und Analyseprozess zum Vergleich mit dem Schwellenwert für die Speicherbelegung in der Tabelle STMG_CURR_THRESHOLD verwendet werden.
DATA_SIZE	BIGINT	J	NICHT VERFÜGBAR
DATA_PAGE	BIGINT	J	NICHT VERFÜGBAR

Tabelle STMG_TABLE:

Die Tabelle STMG_TABLE enthält eine Zeile für jeden detaillierten Eintrag von Tabellenspeichermoment-
aufnahmen.

Tabelle 33. Tabelle STMG_TABLE

Spaltenname	Datentyp	Nullwert?	Beschreibung
STMG_TIMESTAMP	TIMESTAMP	N	Die Zeitmarke der Speichermomentaufnahme. Sie gibt an, wann der Datenerfassungsprozess gestartet wurde.
OBJ_ID	VARCHAR	N	Die eindeutige Kennung für jedes Speicherobjekt unter einer gegebenen Zeitmarke einer Speichermomentaufnahme
COMPLETE_TIMESTAMP	TIMESTAMP	J	Die Zeitmarke für den Zeitpunkt, zu dem der in der Spalte OBJ_ID bezeichnete Datenerfassungsprozess für die Tabelle abgeschlossen wurde
DBPG_NAME	VARCHAR	J	Der Name der Datenbankpartitionsgruppe, in der sich die Tabelle befindet

Tabelle 33. Tabelle STMG_TABLE (Forts.)

Spaltenname	Datentyp	Nullwert?	Beschreibung
SKEW_THRESHOLD_EXCEEDED	SMALLINT	J	Eine Markierung, die den Datenverteilungsstatus der Tabelle angibt 0 = normaler Betriebsstatus; 1 = Warnungsschwellenwert überschritten; 2 = Alarmschwellenwert überschritten
TOTAL_ROW_COUNT	BIGINT	J	Gesamtzeilenzahl der Tabelle
AVG_ROW_COUNT	BIGINT	J	Durchschnittliche Zeilenzahl für alle Tabellenpartitionen
TARGET_LEVEL	BIGINT	J	Die durchschnittliche Datengröße in jeder Partition (in Byte)
DATA_SKEW	SMALLINT	J	Der maximale Prozentwert des Werts ROW_COUNT für die Abweichung vom Wert TARGET_LEVEL für alle Tabellenpartitionen der jeweiligen Tabelle. Dieser Wert wird beim Datenerfassungs- und Analyseprozess zum Vergleich mit dem Schwellenwert für die Datenabweichung in der Tabelle STMG_CURR_THRESHOLD verwendet.
AVG_ROW_LENGTH	BIGINT	J	Die durchschnittliche Zeilenlänge der Tabelle. Wenn diese Statistik erfasst wurde, stellt sie die Summe der durchschnittlichen Spaltenlängen aller Spalten dieser Tabelle dar. Wenn keine Statistikdaten verfügbar sind, wird dieser Wert durch Addieren der festen Spaltenlängen mit dem Prozentsatz der variablen Spaltenlängen berechnet.
COLCOUNT	INTEGER	J	Wie in SYSCAT.TABLES definiert
ESTIMATED_SIZE	BIGINT	J	Wie in SYSCAT.TABLES definiert
NPAGES	INTEGER	J	Wie in SYSCAT.TABLES definiert
FPAGES	INTEGER	J	Wie in SYSCAT.TABLES definiert
OVERFLOW	INTEGER	J	Wie in SYSCAT.TABLES definiert
MAIN_TBSPACE	VARCHAR	J	Wie in SYSCAT.TABLES definiert
INDEX_TBSPACE	VARCHAR	J	Wie in SYSCAT.TABLES definiert
LONG_TBSPACE	VARCHAR	J	Wie in SYSCAT.TABLES definiert

Tabelle STMG_TBPARTITION:

Die Tabelle STMG_TBPARTITION enthält eine Zeile für jeden detaillierten Eintrag von Speichermomentaufnahmen für Tabellenpartitionen.

Tabelle 34. Tabelle STMG_TBPARTITION

Spaltenname	Datentyp	Nullwert?	Beschreibung
STMG_TIMESTAMP	TIMESTAMP	N	Die Zeitmarke der Speichermomentaufnahme. Sie gibt an, wann der Datenerfassungsprozess gestartet wurde.

Tabelle 34. Tabelle STMG_TBPARTITION (Forts.)

Spaltenname	Datentyp	Nullwert?	Beschreibung
OBJ_ID	VARCHAR	N	Die eindeutige Kennung für jedes Speicherobjekt unter einer gegebenen Zeitmarke einer Speichermomentaufnahme
PARTITION_NUM	INTEGER	N	Die Partitionsnummer der Datenbankpartition, in der sich die Tabellenpartition befindet
COMPLETE_TIMESTAMP	TIMESTAMP	J	Die Zeitmarke für den Zeitpunkt, zu dem der in der Spalte OBJ_ID bezeichnete Datenerfassungsprozess für die Tabellenpartition abgeschlossen wurde
DBPG_NAME	VARCHAR	J	Der Name der Datenbankpartitionsgruppe, in der sich die Tabelle befindet
ROWCOUNT	BIGINT	J	Die Anzahl von Zeilen in dieser Tabellenpartition

Tabelle STMG_INDEX:

Die Tabelle STMG_INDEX enthält eine Zeile für jeden detaillierten Eintrag von Indexspeichermomentaufnahmen.

Tabelle 35. Tabelle STMG_INDEX

Spaltenname	Datentyp	Nullwert?	Beschreibung
STMG_TIMESTAMP	TIMESTAMP	N	Die Zeitmarke der Speichermomentaufnahme. Sie gibt an, wann der Datenerfassungsprozess gestartet wurde.
OBJ_ID	VARCHAR	N	Die eindeutige Kennung für jedes Speicherobjekt unter einer gegebenen Zeitmarke einer Speichermomentaufnahme
COMPLETE_TIMESTAMP	TIMESTAMP	J	Die Zeitmarke für den Zeitpunkt, zu dem der in der Spalte OBJ_ID bezeichnete Datenerfassungsprozess für den Index abgeschlossen wurde
DBPG_NAME	VARCHAR	J	Der Name der Datenbankpartitionsgruppe, in der sich der Index befindet
TB_SCHEMA	VARCHAR	J	Wie TABSCHEMA in SYSCAT.INDEXES definiert
TB_NAME	VARCHAR	J	Wie TABNAME SYSCAT.INDEXES definiert
CR_THRESHOLD_EXCEEDED	SMALLINT	J	Eine Markierung, die den Index-Clusterbildungsstatus angibt 0 = normaler Betriebsstatus; 1 = Warnungsschwellenwert überschritten; 2 = Alarmschwellenwert überschritten
COLCOUNT	INTEGER	J	Wie in SYSCAT.INDEXES definiert
ESTIMATED_SIZE	BIGINT	J	Wie in SYSCAT.INDEXES definiert
NLEAF	INTEGER	J	Wie in SYSCAT.INDEXES definiert

Tabelle 35. Tabelle STMG_INDEX (Forts.)

Spaltenname	Datentyp	Nullwert?	Beschreibung
NLEVELS	SMALLINT	J	Wie in SYSCAT.INDEXES definiert
FIRSTKEYCARD	BIGINT	J	Wie in SYSCAT.INDEXES definiert
FIRST2KEYCARD	BIGINT	J	Wie in SYSCAT.INDEXES definiert
FIRST3KEYCARD	BIGINT	J	Wie in SYSCAT.INDEXES definiert
FIRST4KEYCARD	BIGINT	J	Wie in SYSCAT.INDEXES definiert
FULLKEYCARD	BIGINT	J	Wie in SYSCAT.INDEXES definiert
CLUSTERRATIO	SMALLINT	J	Wie in SYSCAT.INDEXES definiert. Dieser Wert wird beim Datenerfassungs- und Analyseprozess zum Vergleich mit dem für den jeweiligen Index festgelegten Schwellenwert verwendet.
CLUSTERFACTOR	BIGINT	J	Wie in SYSCAT.INDEXES definiert
SEQUENTIAL_PAGES	INTEGER	J	Wie in SYSCAT.INDEXES definiert
DENSITY	INTEGER	J	Wie in SYSCAT.INDEXES definiert

Zugehörige Referenzen:

• „Speicherverwaltungssicht" auf Seite 133

Entwurf temporärer Tabellenbereiche

Es wird empfohlen, einen einzelnen temporären SMS-Tabellenbereich zu definieren, dessen Seitengröße der Seitengröße entspricht, die in den meisten regulären Tabellenbereichen verwendet wird. Dies sollte für typische Umgebungen und Auslastungen geeignet sein. Es kann jedoch vorteilhaft sein, mit unterschiedlichen Konfigurationen für temporäre Tabellenbereiche und Auslastungen zu experimentieren. Sie sollten die folgenden Punkte beachten:

- Auf temporäre Tabellen wird meist gruppenweise und sequenziell zugegriffen. Das heißt, eine Gruppe von Zeilen wird eingefügt oder eine Gruppe sequenzieller Zeilen wird abgerufen. Daher führt eine größere Seitengröße in der Regel zu einer besseren Leistung, weil weniger E/A-Anforderungen logischer oder physischer Seiten erforderlich sind, um eine bestimmte Datenmenge einzulesen. Dies ist nicht immer der Fall, wenn die durchschnittliche Zeilengröße einer temporären Tabelle kleiner ist als die Seitengröße dividiert durch 255. Ungeachtet der Seitengröße können maximal 255 Zeilen auf einer Seite vorhanden sein. Eine Abfrage, die z. B. eine temporäre Tabelle mit 15-Byte-Zeilen erfordert, könnte von einem temporären Tabellenbereich mit einer Seitengröße von 4 KB besser bedient werden, da alle 255 solcher Zeilen in eine 4-KB-Seite aufgenommen werden können. Eine 8-KB-Seite (oder größere Seite) ergäbe mindestens 4 KB (oder mehr) Byte ungenutzten Speicherbereich auf jeder Seite der temporären Tabelle und würde die Anzahl der erforderlichen E/A-Anforderungen nicht reduzieren.
- Wenn über fünfzig Prozent der regulären Tabellenbereiche in der Datenbank dieselbe Seitengröße verwenden, kann es vorteilhaft sein, die temporären Tabellenbereiche mit derselben Seitengröße zu definieren. Der Vorteil liegt darin, dass der temporäre Tabellenbereich mit dieser Konfiguration denselben Pufferpoolspeicherbereich mit den meisten oder allen regulären Tabellenbereichen gemeinsam benutzen kann. Dadurch wird die Optimierung des Pufferpools wiederum vereinfacht.

- Wenn Sie mit Hilfe eines temporären Tabellenbereichs eine Tabelle reorganisieren, muss die Seitengröße des temporären Tabellenbereichs mit der Seitengröße der Tabelle übereinstimmen. Deshalb sollten Sie sicherstellen, dass temporäre Tabellenbereiche vorhanden sind, die für jede Seitengröße definiert sind, die von vorhandenen Tabellen verwendet wird, die Sie vielleicht mit Hilfe eines temporären Tabellenbereichs reorganisieren.

  Sie können eine Reorganisation auch ohne temporären Tabellenbereich ausführen, indem Sie die Tabelle direkt im Zieltabellenbereich reorganisieren. Natürlich setzt diese Art der Reorganisation voraus, dass im Zieltabellenbereich zusätzlicher Speicherbereich für den Reorganisationsprozess vorhanden ist.
- Wenn Sie aufgrund Ihrer Arbeitsumgebung temporäre Systemtabellen in SMS-Tabellenbereichen für temporäre Systemtabellen verwenden, können Sie die Verwendung der Registriervariablen DB2_SMS_TRUNC_TMPTABLE_THRESH in Betracht ziehen. Wenn in der Vergangenheit temporäre Systemtabellen nicht mehr benötigt wurden, wurden sie auf die Dateigröße null abgeschnitten. Wenn eine neue temporäre Systemtabelle erstellt werden musste, war dies mit Leistungsaufwand verbunden. Durch die Verwendung dieser Registriervariablen können nun temporäre Systemtabellen mit einer anderen Größe als null im System zurückbehalten werden, um den Aufwand wiederholter Erstellungen und Abschneidungen temporärer Systemtabellen zu vermeiden.
- Im Allgemeinen wählt das Optimierungsprogramm, wenn temporäre Tabellenbereiche mit unterschiedlichen Seitengrößen vorhanden sind, am häufigsten den temporären Tabellenbereich mit dem größten Pufferpool aus. In solchen Fällen empfiehlt es sich, einem der temporären Tabellenbereiche einen großen Pufferpool und den übrigen einen kleineren Pufferpool zuzuordnen. Eine solche Pufferpoolzuordnung hilft bei der Sicherstellung einer effizienten Auslastung des Hauptspeichers. Wenn z. B. Ihr Katalogtabellenbereich 4-KB-Seiten verwendet und die übrigen Tabellenbereiche 8-KB-Seiten, kann sich folgende Konfiguration für temporäre Tabellenbereiche am besten eignen: ein einzelner temporärer 8-KB-Tabellenbereich mit einem großen Pufferpool und ein einzelner 4-KB-Tabellenbereich mit einem kleinen Pufferpool.

  Anmerkung: Katalogtabellenbereiche sind auf die Verwendung von 4-KB-Seiten beschränkt. Daher erzwingt der Datenbankmanager immer das Vorhandensein eines 4-KB-Tabellenbereichs für temporäre Systemtabellen, um die Reorganisation von Katalogtabellen zu ermöglichen.
- Es bringt im Allgemeinen keinen Vorteil, mehr als einen temporären Tabellenbereich von jeder Seitengröße zu definieren.
- Für temporäre Tabellenbereiche sind SMS-Bereiche aus den folgenden Gründen DMS-Bereichen meist vorzuziehen:
  - Im Vergleich zu SMS-Tabellenbereichen erfordert die Erstellung einer temporären Tabelle in DMS-Tabellenbereichen mehr Systemaufwand.
  - Plattenspeicherplatz wird im SMS nach Bedarf zugeordnet, wohingegen er im DMS zuvor zugeordnet werden muss. Eine vorherige Zuordnung kann ein Problem darstellen: Temporäre Tabellenbereiche enthalten Übergangsdaten, die einen extremen Höchstspeicherbedarf und einen wesentlich geringeren durchschnittlichen Speicherbedarf besitzen können. Bei DMS muss der Höchstspeicherbedarf vorab zugeordnet werden, wohingegen bei SMS der zusätzliche Plattenspeicherplatz außerhalb der Spitzenlastzeiten für andere Zwecke verwendet werden kann.
  - Der Datenbankmanager versucht, temporäre Tabellenseiten im Speicher zu behalten und sie nicht auf die Platte auszulagern. Im Endeffekt sind die Leistungsvorteile von DMS weniger signifikant.

Zugehörige Konzepte:

- „Aufbau von Tabellenbereichen" auf Seite 100
- „Vom Betriebssystem verwalteter Speicherbereich (SMS)" auf Seite 104
- „Temporäre Tabellen in SMS-Tabellenbereichen" auf Seite 147

Zugehörige Referenzen:

- „REORG INDEXES/TABLE Command" in *Command Reference*

Temporäre Tabellen in SMS-Tabellenbereichen

Temporäre Tabellen in SMS-Tabellenbereichen werden nicht standardmäßig gelöscht, wenn sie nicht mehr benötigt werden. Stattdessen werden Dateien, die temporären Tabellen zugeordnet sind, die größer als eine EXTENTSIZE-Größe sind, auf eine EXTENTSIZE-Größe abgeschnitten. In Fällen, in denen temporäre Tabellen wiederholt verwendet werden, vermeidet dieses Verfahren einigen Leistungsaufwand zum Löschen und erneuten Erstellen temporärer Tabellen.

Diese Wiederverwendung temporärer Tabellen bietet Vorteile für Benutzer, deren Auslastung mit vielen kleinen temporären Tabellen auf kleineren Systemen wie Windows® NT, unter denen die Dateisystemaufrufe relativ aufwendig sind, verbunden ist, sowie für Benutzer, deren Plattenspeicher verteilt ist, so dass Netznachrichten zur Ausführung der Dateisystemoperationen erforderlich sind.

Standardmäßig werden Dateien, die temporäre Tabellen enthalten, die größer als eine EXTENTSIZE-Größe sind, auf eine EXTENTSIZE-Größe abgeschnitten, wenn sie nicht mehr benötigt werden. Sie können diesen Betrag erhöhen, indem Sie einen größeren Wert für die Registriervariable DB2_SMS_TRUNC_TMPTABLE_THRESH angeben. Sie sollten den Wert, der dieser Registriervariablen zugeordnet ist, erhöhen, wenn Ihre Auslastung wiederholt große temporäre SMS-Tabellen verwendet und Sie es sich leisten können, den Speicherplatz zwischen den Verwendungen zugeordnet zu lassen.

Sie können diese Funktion ausschalten, indem Sie den Wert 0 für die Registriervariable DB2_SMS_TRUNC_TMPTABLE_THRESH angeben. Dies kann sinnvoll sein, wenn Ihr System über sehr begrenzte Speicherressourcen verfügt und Sie wiederholt auf Fehler wegen Plattenspeichermangels für temporäre SMS-Tabellenbereiche gestoßen sind.

Die erste Verbindung zur Datenbank löscht alle zuvor zugeordneten Dateien. Wenn Sie alle vorhandenen temporären Tabellen löschen wollen, sollten Sie alle Datenbankverbindungen beenden und eine neue Verbindung herstellen oder die Datenbank inaktivieren und erneut aktivieren. Wenn Sie sicherstellen wollen, dass der Speicher für temporäre Tabellen zugeordnet bleibt, verwenden Sie den Befehl ACTIVATE DATABASE, um die Datenbank zu starten. Dadurch wird der wiederholte Startaufwand beim Herstellen der ersten Verbindung zur Datenbank vermieden.

Zugehörige Konzepte:

- „Entwurf temporärer Tabellenbereiche" auf Seite 145

Entwurf von Katalogtabellenbereichen

Ein SMS-Tabellenbereich empfiehlt sich aus folgenden Gründen für Datenbankkataloge:

- Der Datenbankkatalog besteht aus zahlreichen Tabellen unterschiedlicher Größen. Bei der Verwendung eines DMS-Tabellenbereichs wird mindestens ein Speicherbereich in der Größe von zwei EXTENTSIZE-Bereichen für jedes Tabellenobjekt zugeordnet. Je nachdem, welcher Wert für EXTENTSIZE ausgewählt wurde, kann dies zu einer erheblichen Menge an zugeordnetem, aber ungenutztem Speicher führen. Wenn ein DMS-Tabellenbereich verwendet wird, sollte ein kleiner Wert für EXTENTSIZE (zwei bis vier Seiten) ausgewählt werden. Ansonsten sollte ein SMS-Tabellenbereich verwendet werden.
- In den Katalogtabellen gibt es Spalten mit großen Objekten (LOB-Spalten). LOB-Daten werden nicht mit anderen Daten im Pufferpool behalten, sondern jedes Mal, wenn sie benötigt werden, von der Platte gelesen. Das Lesen von LOB-Daten von der Platte verlangsamt die Leistung. Da ein Dateisystem in der Regel über einen eigenen Cache verfügt, kann die Verwendung eines SMS-Tabellenbereichs oder eines DMS-Tabellenbereichs, der aus Dateibehältern besteht, die E/A-Operationen möglicherweise umgehen, wenn auf die LOB-Daten zuvor bereits zugegriffen wurde.

Unter diesen Gesichtspunkten erweist sich ein SMS-Tabellenbereich als die etwas günstigere Wahl für die Katalogtabellen.

Ein anderer Faktor, der zu berücksichtigen wäre, ist die Frage, ob der Katalogtabellenbereich in der Zukunft erweitert werden müsste. Obwohl einige Plattformen die Erweiterung des zugrundeliegenden Speichers für SMS-Behälter unterstützen und die Möglichkeit einer umgeleiteten Wiederherstellung zur Vergrößerung eines SMS-Tabellenbereichs gegeben ist, macht ein DMS-Tabellenbereich das Hinzufügen neuer Behälter einfacher.

Zugehörige Konzepte:

- „Definition von Systemkatalogtabellen" in *Systemverwaltung: Implementierung*
- „Aufbau von Tabellenbereichen" auf Seite 100
- „Vom Betriebssystem verwalteter Speicherbereich (SMS)" auf Seite 104
- „Von der Datenbank verwalteter Speicherbereich (DMS)" auf Seite 106

Optimieren der Leistung von Tabellenbereichen bei Datenspeicherung auf RAID-Einheiten

In diesem Abschnitt wird beschrieben, wie Sie die Leistung optimieren können, wenn Daten auf RAID-Einheiten (Redundant Array of Independent Disks) gespeichert werden.

Vorgehensweise:

Sie sollten folgende Maßnahmen für jeden Tabellenbereich ergreifen, der eine RAID-Einheit verwendet:

- Definieren Sie einen einzelnen Behälter für den Tabellenbereich (der eine RAID-Einheit verwendet).

- Setzen Sie den Wert für EXTENTSIZE des Tabellenbereichs so, dass er der Größe der einheitenübergreifend gespeicherten RAID-Datenblöcke (Stripe Size) oder einem Vielfachen der Größe entspricht.
- Stellen Sie sicher, dass der Wert für PREFETCHSIZE des Tabellenbereichs die folgenden Bedingungen erfüllt:
  - Er entspricht der Größe der einheitenübergreifend gespeicherten RAID-Datenblöcke (Stripe Size) multipliziert mit der Anzahl paralleler RAID-Einheiten (oder einem ganzen Vielfachen dieses Produkts).
  - Er ist ein Vielfaches des Werts für EXTENTSIZE.
- Verwenden Sie die Registriervariable DB2_PARALLEL_IO, um die parallele Ein-/Ausgabe für den Tabellenbereich zu aktivieren.

DB2_PARALLEL_IO:

Wenn Daten aus Tabellenbereichsbehältern gelesen werden oder in diese geschrieben werden, kann DB2 Universal Database™ (DB2 UDB) die parallele Ein-/Ausgabe verwenden, falls die Anzahl der Behälter in der Datenbank größer als 1 ist. Es gibt jedoch Fälle, in denen es vorteilhaft wäre, die parallele Ein-/Ausgabe für einzelne Tabellenbereichsbehälter zu aktivieren. Wenn z. B. der Behälter auf einer einzelnen RAID-Einheit erstellt wird, die aus mehr als einer physischen Platte besteht, kann es sinnvoll sein, Aufrufe zum parallelen Lesen und Schreiben absetzen zu können.

Sie können die Registriervariable DB2_PARALLEL_IO verwenden, um die parallele Ein-/Ausgabe für einen Tabellenbereich zu erzwingen, der einen einzelnen Behälter besitzt. Diese Variable kann auf "*" (Stern) gesetzt werden, d. h. sich auf jeden Tabellenbereich beziehen, oder sie kann auf eine Liste von Tabellenbereichs-IDs gesetzt werden, die durch Komma voneinander getrennt sind. Zum Beispiel:

```
db2set DB2_PARALLEL_IO=*        {parallele E/A für alle
                                 Tabellenbereiche aktivieren}
db2set DB2_PARALLEL_IO=1,2,4,8  {parallele E/A für die Tabellenbereiche 1, 2,
                                 4 und 8 aktivieren}
```

Nach dem Setzen der Registriervariablen muss DB2 UDB gestoppt (**db2stop**) und anschließend erneut gestartet (**db2start**) werden, damit die Änderungen wirksam werden.

DB2_PARALLEL_IO wirkt sich außerdem auf Tabellenbereiche mit mehr als einem definierten Behälter aus. Wenn Sie die Registiervariable nicht definieren, ist die E/A-Parallelität gleich der Anzahl von Behältern im betreffenden Tabellenbereich. Wenn Sie die Registriervariable definieren, ist die E/A-Parallelität gleich dem Quotienten aus dem Wert für PREFETCHSIZE geteilt durch den Wert für EXTENTSIZE. Es kann sinnvoll sein, die Registriervariable zu definieren, wenn die einzelnen Behälter in dem Tabellenbereich über mehrere physische Datenträger einheitenübergreifend verteilt gespeichert werden (Stripeset).

Betrachten Sie zum Beispiel den Fall, dass ein Tabellenbereich zwei Behälter hat und der Wert für PREFETCHSIZE das Vierfache des Werts für EXTENTSIZE beträgt. Wenn die Registriervariable nicht definiert ist, wird eine Vorableseanforderung für diesen Tabellenbereich in zwei Anforderungen zerlegt (jeweils eine Anforderung für zwei EXTENTSIZE große Speicherbereiche). Unter der Voraussetzung, dass die Vorablesefunktionen einsatzbereit zur Verfügung stehen, können zwei Vorablesefunktionen diese Anforderungen parallel bearbeiten. In dem Fall, dass die Registriervariable definiert ist, wird eine Vorableseanforderung für diesen Tabellenbereich in vier Anforderungen zerlegt (ein EXTENTSIZE großer Speicher-

bereich pro Anforderung), was die Möglichkeit bietet, dass vier Vorablesefunktionen die Anforderungen parallel bedienen.

Wenn in diesem Beispiel für jeden der beiden Behälter ein einziger dedizierter Datenträger vorhanden wäre, kann eine Definition der Registriervariablen für diesen Tabellenbereich zu Konkurrenzsituationen beim Zugriff auf diese Datenträger führen, da jeweils zwei Vorablesefunktionen versuchen würden, gleichzeitig auf jeden der beiden Datenträger zuzugreifen. Wenn hingegen jeder der beiden Behälter über mehrere Datenträger einheitenübergreifend gespeichert wäre, würde eine Definition der Registriervariablen potenziell einen gleichzeitigen Zugriff auf vier verschiedene Datenträger ermöglichen.

DB2_USE_PAGE_CONTAINER_TAG:

Standardmäßig speichert DB2 UDB im ersten EXTENTSIZE-Bereich jedes DMS-Behälters (Datei- oder Einheitenbehälters) eine Behälterkennung (container tag). Die Behälterkennung stellt die Metadaten von DB2 UDB für den Behälter dar. In früheren Versionen von DB2 UDB wurde anstelle des ersten EXTENTSIZE-Bereichs die erste Seite für die Behälterkennung und dadurch weniger Speicherbereich zum Speichern der Kennung verwendet. (In früheren DB2 UDB-Versionen diente die Registriervariable DB2_STRIPED_CONTAINERS zur Erstellung von Tabellenbereichen mit einer EXTENTSIZE großen Behälterkennung. Da dies jedoch nun das Standardverfahren ist, hat diese Registriervariable keine Wirkung mehr.)

Wenn die Registriervariable DB2_USE_PAGE_CONTAINER_TAG auf den Wert ON gesetzt ist, wird jeder neue DMS-Behälter mit einer Behälterkennung in der Größe einer Seite anstatt eines EXTENTSIZE-Bereichs (des Standardwerts) erstellt. Auf vorhandene Behälter, die vor der Definition der Registriervariablen erstellt wurden, hat dies jedoch keine Auswirkung.

Die Einstellung dieser Registriervariablen auf den Wert ON wird nicht empfohlen, sofern Sie nicht unter sehr eingeschränkten Speicherverhältnissen arbeiten oder eine Funktionsweise benötigen, die mit der früherer Versionen (vor Version 8) übereinstimmt.

Der Wert ON für diese Registriervariable kann sich negativ auf die E/A-Leistung auswirken, wenn RAID-Einheiten für Tabellenbereichsbehälter verwendet werden. Wenn RAID-Einheiten für Tabellenbereichsbehälter verwendet werden, wird empfohlen, dass der Tabellenbereich mit einem Wert für EXTENTSIZE erstellt wird, der der Größe der einheitenübergreifend gespeicherten RAID-Datenblöcke (Stripe Size) oder einem Vielfachen dieser Größe entspricht. Wenn diese Registriervariable jedoch auf den Wert ON gesetzt wird, wird eine Behälterkennung in der Größe einer Seite verwendet und die EXTENTSIZE-Bereiche richten sich nicht nach den einheitenübergreifend gespeicherten RAID-Datenblöcken (Stripes) aus. Infolgedessen kann bei einer E/A-Anforderung ein Zugriff auf mehr als die optimale Anzahl physischer Platten erforderlich werden. Benutzern wird daher dringend empfohlen, diese Registriervariable nicht zu definieren.

Wenn Sie Behälter mit Behälterkennungen von der Größe einer Seite erstellen wollen, setzen Sie diese Registriervariable auf den Wert ON und stoppen und starten das Exemplar erneut:

```
db2set DB2_USE_PAGE_CONTAINER_TAG=ON
db2stop
db2start
```

Sie können das Erstellen von Behältern mit Behälterkennungen von der Größe einer Seite unterbinden, indem Sie die Registriervariable zurücksetzen und anschließend das Exemplar stoppen und erneut starten.

```
db2set DB2_USE_PAGE_CONTAINER_TAG=
db2stop
db2start
```

Die Steuerzentrale und der Befehl LIST TABLESPACE CONTAINERS zeigen nicht an, ob ein Behälter mit einer Behälterkennung von der Größe einer Seite oder eines EXTENTSIZE-Bereichs erstellt wurde. Sie verwenden die Bezeichnung „Datei" oder „Einheit", abhängig davon, wie der Behälter erstellt wurde. Um zu prüfen, ob ein Behälter mit einer Kennung in der Größe einer Seite oder des EXTENTSIZE-Werts erstellt wurde, können Sie die Option /DTSF des Befehls DB2DART verwenden, um Informationen zu Tabellenbereichen und Behältern auszulesen, und sich dann das Typfeld für den fraglichen Behälter ansehen. Mit Hilfe der APIs zum Abfragen von Behältern (sqlbftcq und sqlbtcq) lässt sich eine einfache Anwendung erstellen, die den Typ anzeigt.

Zugehörige Konzepte:

- „Aufbau von Tabellenbereichen" auf Seite 100

Zugehörige Referenzen:

- „Systemumgebungsvariablen" in *Systemverwaltung: Optimierung*

Überlegungen zur Auswahl von Tabellenbereichen für Tabellen

Bei der Festlegung, wie Tabellen Tabellenbereichen zugeordnet werden sollen, sollten Sie Folgendes berücksichtigen:

- Die Partitionierung Ihrer Tabellen

  Sie sollten mindestens sicherstellen, dass sich der Tabellenbereich, den sie auswählen, in einer Datenbankpartitionsgruppe mit der gewünschten Partitionierung befindet.

- Menge der Daten in der Tabelle

  Wenn Sie planen, viele kleine Tabellen in einem Tabellenbereich zu speichern, sollten Sie dazu die Verwendung eines SMS-Tabellenbereichs in Betracht ziehen. Die Vorteile von DMS-Tabellenbereichen im Hinblick auf die Effizienz bei E/A-Operationen und Speicherbereichsverwaltung sind bei kleinen Tabellen nicht so bedeutsam. Die Vorteile von SMS-Tabellenbereichen hinsichtlich der Speicherzuordnung von jeweils einer Seite und nur bei Bedarf sind bei kleinen Tabellen attraktiver. Wenn eine Ihrer Tabellen größer ist oder Sie einen schnelleren Zugriff auf die Daten in den Tabellen benötigen, sollte ein DMS-Tabellenbereich mit einem kleinen Wert für EXTENTSIZE in Betracht gezogen werden.

  Eventuell empfiehlt es sich, einen separaten Tabellenbereich für jede umfangreiche Tabelle zu verwenden und kleine Tabellen gemeinsam in einem einzigen Tabellenbereich anzulegen. Diese Trennung ermöglicht Ihnen, anhand der Auslastung des Tabellenbereichs einen geeigneten Wert für den Parameter EXTENTSIZE auszuwählen.

- Typ der Daten in der Tabelle

  Sie könnten beispielsweise über Tabellen mit alten Daten verfügen, die relativ selten verwendet werden und bei denen der Endbenutzer eine längere Antwortzeit für Abfragen, die diese Daten betreffen, vielleicht akzeptiert. In diesem Fall könnten Sie für diese „historischen" Tabellen einen anderen Tabellenbereich ver-

wenden und diesem Tabellenbereich kostensparendere physische Einheiten mit einer langsameren Zugriffsgeschwindigkeit zuordnen.

Auf der anderen Seite können Sie einige wichtige Tabellen identifizieren, für die eine schnelle Verfügbarkeit und schnelle Antwortzeiten erforderlich sind. Diese Tabellen könnten Sie in einen Tabellenbereich stellen, der einer schnellen physischen Einheit zugeordnet ist, die die Anforderungen für diese wichtigen Daten besser erfüllt.

Wenn Sie mit DMS-Tabellenbereichen arbeiten, können Sie Ihre Tabellendaten auf drei verschiedene Tabellenbereiche verteilen: einen für Indexdaten, einen für LOB- und Langfelddaten sowie einen für reguläre Tabellendaten. Auf diese Weise können Sie die Tabellenbereichsmerkmale und die physischen Einheiten der Tabellenbereiche auswählen, die für die Daten am besten geeignet sind. Sie könnten z. B. die Indexdaten auf die schnellste verfügbare Einheit stellen und dadurch eine erhebliche Verbesserung der Leistung erzielen. Wenn Sie eine Tabelle auf DMS-Tabellenbereiche aufteilen, sollten Sie in Betracht ziehen, diese Tabellenbereiche zusammen zu sichern und wiederherzustellen, wenn die aktualisierende Wiederherstellung (Rollforward) aktiviert ist. SMS-Tabellenbereiche unterstützen diese Art der Datenverteilung auf Tabellenbereiche nicht.

- Verwaltungserfordernisse

  Einige Verwaltungsfunktionen können auf Tabellenbereichsebene anstatt auf Datenbank- oder Tabellenebene ausgeführt werden. Wenn Sie beispielsweise eine Sicherungskopie eines Tabellenbereichs anstelle der Sicherungskopie einer Datenbank erstellen, können Sie Ihre Zeit und Ressourcen besser ausnutzen. Diese Vorgehensweise ermöglicht Ihnen, Tabellenbereiche mit umfangreichen Änderungen häufig zu sichern und von den Tabellenbereichen, die wenig geändert werden, nur gelegentlich neue Sicherungskopien anzulegen.

  Sie können eine Datenbank oder einen Tabellenbereich wiederherstellen. Wenn Tabellen, die sich nicht aufeinander beziehen, keine gemeinsamen Tabellenbereiche benutzen, haben Sie die Option, kleinere Teile Ihrer Datenbank wiederherzustellen und den Aufwand verringern.

  Ein gutes Verfahren besteht darin, Tabellen, die voneinander abhängig sind, in einer Gruppe von Tabellenbereichen zusammenzufassen. Diese Tabellen könnten über referenzielle Integritätsbedingungen oder andere definierte Geschäftsregeln voneinander abhängig sein.

  Falls Sie eine bestimmte Tabelle häufig löschen und erneut definieren müssen, können Sie die Tabelle in einem eigenen Tabellenbereich definieren, weil das Löschen eines DMS-Tabellenbereichs effizienter ist als das Löschen einer Tabelle.

Zugehörige Konzepte:

- „Datenbankpartitionsgruppen" auf Seite 91
- „Vom Betriebssystem verwalteter Speicherbereich (SMS)" auf Seite 104
- „Von der Datenbank verwalteter Speicherbereich (DMS)" auf Seite 106
- „SMS- und DMS-Tabellenbereiche im Vergleich" auf Seite 125

In DB2 UDB verwendete Tabellen

DB2® Universal Database (DB2 UDB) stellt die folgenden Tabellentypen bereit:

- Reguläre Tabellen, die als Zwischenspeicher implementiert sind.
- Tabellen für den Anfügemodus, bei denen es sich um reguläre Tabellen handelt, die primär für INSERT-Operationen optimiert wurden.

- MDC-Tabellen (MDC = Multidimensional Clustering), die als Tabellen implementiert sind, für die eine physische Clusterbeziehung mit mehr als einem Schlüssel (bzw. mehr als einer Dimension) gleichzeitig definiert ist.
- Bereichsclustertabellen (RCT = Range-Clustered Table), die als sequenzielle Datencluster implementiert sind und schnellen, direkten Zugriff bieten.

Jeder Tabellentyp verfügt über Merkmale, die den Einsatz in einer bestimmten Unternehmensumgebung sinnvoll machen. Bei jeder verwendeten Tabelle sollten Sie prüfen, welcher Tabellentyp für Ihre Anforderungen am besten geeignet ist.

Reguläre Tabellen mit Indizes sind die „allgemein einsetzbaren" Tabellen.

Reguläre Tabellen werden durch eine Anweisung ALTER TABLE in den Anfügemodus versetzt. Tabellen für den Anfügemodus eignen sich, wenn Sie neue Daten hinzufügen und vorhandene Daten abrufen müssen. Dieser Tabellentyp kann z. B. im Bankenwesen für die Bearbeitung von Kundenkonten verwendet werden. In diesen Tabellen können dann alle Kontenbewegungen durch Gut- und Lastschriften sowie Überweisungen aufgezeichnet werden. Über diese Tabellen können Sie auch Daten für Kunden bereitstellen, die den Verlauf der Kontenbewegungen prüfen wollen.

MDC-Tabellen werden beim Data Warehousing und in umfangreichen Datenbankumgebungen verwendet. Clusterindizes in regulären Tabellen unterstützen das eindimensionale Clustering von Daten. MDC-Tabellen bieten die Vorteile des Datenclusterings in mehr als einer Dimension.

Bereichsclustertabellen werden verwendet, wenn die Daten eine dichte Clusterbildung über eine oder mehrere Tabellenspalten aufweisen. Die höchsten und niedrigsten Werte in den Spalten definieren den Bereich der möglichen Werte. Diese Spalten werden für den Zugriff auf die Datensätze der Tabelle verwendet.

Zugehörige Konzepte:

- „Tabellen mit mehrdimensionalem Clustering" auf Seite 158
- „Bereichsclustertabellen" auf Seite 153

Zugehörige Tasks:

- „Erstellen und Füllen einer Tabelle" in *Systemverwaltung: Implementierung*
- „Erstellen einer gespeicherten Abfragetabelle" in *Systemverwaltung: Implementierung*

Bereichsclustertabellen

Bei einer Bereichsclustertabelle (RCT = Range-Clustered Table) handelt es sich um ein Tabellenlayoutschema, in dem jeder Datensatz in der Tabelle eine vorbestimmte Satz-ID (RID) besitzt, die eine interne Kennung zur Lokalisierung eines Datensatzes in einer Tabelle darstellt.

Bei jeder Tabelle, die Ihre Daten enthält, sollten Sie prüfen, welcher Tabellentyp für Ihre Anforderungen am besten geeignet ist. Wenn Sie z. B. mit Datensätzen arbeiten, die über eine lockere Clusterbildung verfügen (ohne monotones Wachstum), sollten Sie prüfen, ob reguläre Tabellen mit Indizes eingesetzt werden können. Wenn Sie über Datensätze verfügen, die doppelte (nicht eindeutige) Werte im Schlüssel aufweisen, sollten Sie keine Bereichsclustertabellen verwenden. Wenn Sie vorab keine festgelegte Menge an Plattenspeicherplatz für Bereichsclustertabellen

zuordnen können, sollten Sie diesen Tabellentyp auch nicht verwenden. Diese Faktoren erleichtern Ihnen die Feststellung, ob Sie über Daten verfügen, die sich für eine Bereichsclustertabelle eignen.

Der Wert des Datensatzschlüssels wird über einen Algorithmus mit der Speicherposition einer bestimmten Tabellenzeile innerhalb einer Tabelle gleichgesetzt. Der Basisalgorithmus ist hierbei relativ einfach. In seiner Grundform (bei der sich der Schlüssel aus einer einzigen Spalte und nicht aus zwei oder mehr Spalten zusammensetzt) ordnet dieser Algorithmus einer Folgenummer eine logische Zeilennummer zu. Der Algorithmus verwendet außerdem den Schlüssel des Datensatzes, um die logische Seiten- und Bereichsnummer festzustellen. Dieser Prozess bietet einen außerordentlich schnellen Zugriff auf Datensätze, d. h. auf bestimmte Zeilen innerhalb der Tabelle.

Der Algorithmus enthält keine Hashoperationen, da bei einem Hashverfahren die Schlüssel/Wert-Abfolge nicht erhalten bleibt. Die Beibehaltung der definierten Schlüssel/Wert-Abfolge ist jedoch von großer Bedeutung, da durch sie die Notwendigkeit zur Reorganisation der Tabelle entfällt.

Jeder Datensatzschlüssel in der Tabelle muss über die folgenden Merkmale verfügen:

- Eindeutigkeit
- Wert ungleich Null
- Integerwert (SMALLINT, INTEGER oder BIGINT)
- Monoton wachsend
- Wert innerhalb einer vordefinierten Bereichsgruppe auf der Basis der einzelnen Schlüsselspalten

  Die Option ALLOW OVERFLOW wird bei der Tabellenerstellung verwendet, wenn die Schlüsselwerte den definierten Bereich übersteigen dürfen. Die Option DISALLOW OVERFLOW wird bei der Tabellenerstellung verwendet, wenn die Schlüsselwerte den definierten Bereich nicht übersteigen dürfen. In diesem Fall wird eine SQL-Fehlernachricht ausgegeben, wenn ein Datensatz eingefügt wird, der den zulässigen Bereich übersteigt.

Anwendungen, bei denen Folgeschlüsselbereiche mit dichter Clusterbildung wahrscheinlich sind, bieten sich hervorragend für Bereichsclustertabellen an. Wenn Sie diesen Schlüsseltyp zum Erstellen einer Bereichsclustertabelle verwenden, wird der Schlüssel zum Generieren der logischen Speicherposition einer Zeile innerhalb einer Tabelle benutzt. Durch diesen Prozess ist kein separater Index mehr erforderlich.

Die Verwendung von Bereichsclustertabellen bietet die folgenden Vorteile:

- Direktzugriff

  Der Zugriff erfolgt über eine Bereichsclustertabellen-Funktion zur Zuordnung eines Schlüssels zu einer RID.
- Weniger Wartungsaufwand

  Eine Sekundärstruktur wie z. B. eine B+-Baumstruktur muss nicht nach jeder INSERT-, UPDATE- oder DELETE-Operation aktualisiert werden.
- Weniger Protokollierungsoperationen

  Im Vergleich zu etwa gleich großen regulären Tabellen und den zugehörigen B+-Indexstrukturen ist bei Bereichsclustertabellen ein geringerer Protokollierungsaufwand erforderlich.

- Weniger Pufferpoolspeicher

  Es wird kein zusätzlicher Speicherbereich zum Speichern von Sekundärstrukturen benötigt.
- Reihenfolge von B+-Baumstrukturtabellen

  Die Anordnung der Datensätze ist mit der Anordnung in B+-Baumstrukturtabellen identisch, ohne dass zusätzliche Ebenen oder B+-Baumstrukturschemata zur Sperrung des nächsten Schlüssels erforderlich sind. Bei Bereichsclustertabellen wird die Codepfadlänge im Vergleich zu regulären B+-Indexstrukturen reduziert. Zur Nutzung dieses Vorteils muss die Bereichsclustertabelle allerdings mit der Option DISALLOW OVERFLOW erstellt worden sein, und die Daten müssen eine dichte Clusterbildung aufweisen.
- Weniger Indizes

  Durch die Zuordnung jedes Schlüssels zu einer Plattenposition wird bei der Tabellenerstellung ein Index weniger benötigt als dies sonst erforderlich wäre. Bei Bereichsclustertabellen ist auf Grund der Anwendungserfordernisse für den Datenzugriff in der Tabelle ein zweiter, separater Index möglicherweise nicht mehr erforderlich. Sie können jedoch weiterhin reguläre Indizes erstellen, insbesondere wenn die Anwendung dies erforderlich macht.

Indizes werden zur Ausführung folgender Funktionen verwendet:

- Lokalisieren eines Datensatzes auf der Basis eines Datensatzschlüssels
- Anwenden von Start- und Stoppschlüssel-Suchoperationen
- Ausführen einer vertikalen Partitionierung

  Bei Verwendung von Bereichsclustertabellen wird bei Indizes nur die vertikale Partitionierung nicht berücksichtigt.

Bei der Entscheidung zur Verwendung von Bereichsclustertabellen sollten Sie die folgenden Merkmale berücksichtigen, durch die sich diese Tabellen von regulären Tabellen unterscheiden:

- Bereichsclustertabellen verfügen nicht über Steuersätze für freie Speicherbereiche.
- Der Speicherbereich wird vorab zugeordnet.

  Der Tabelle wird vorab Speicherbereich zugeordnet, der für die Verwendung durch diese Tabelle auch dann reserviert wird, wenn in die Tabelle keine Datensätze eingefügt werden. Bei der Tabellenerstellung sind in der Tabelle keine Datensätze vorhanden. Der gesamte Seitenbereich ist allerdings vorab zugeordnet. Das vorab ausgeführte Zuordnen basiert auf der Datensatzlänge und der maximalen Anzahl der zu speichernden Datensätze.
  - Wenn Felder variabler Länge wie z. B. VARCHAR im Datensatz verwendet werden, wird die maximale Feldlänge verwendet und die Gesamtdatensatzlänge ist fest. Die feste Gesamtlänge der verschiedenen Datensätze wird zusammen mit der maximalen Anzahl der Datensätze zur Feststellung der Speicherplatzanforderung verwendet.
  - Dies kann dazu führen, dass zusätzlicher Speicherplatz zugeordnet wird, der effektiv nicht genutzt werden kann.
  - Wenn zwischen Schlüsselwerten große Lücken liegen, wird Speicherplatz nicht genutzt und die Leistung von Bereichssuchen ist gering.
  - Bereichssuchen müssen alle möglichen Datensätze innerhalb eines Bereichs aufsuchen, selbst wenn die Zeilen, die die entsprechenden Schlüsselwerte enthalten, noch nicht in die Datenbank eingefügt wurden.

- Schemamodifikationen sind nicht zulässig.

  Wenn eine Schemamodifikation an einer Bereichsclustertabelle erforderlich ist, muss die Tabelle erneut erstellt werden, so dass sie den neuen Schemanamen für die Tabelle und alle Daten aus der alten Tabelle enthält. Insbesondere gilt Folgendes:

  – Das Ändern eines Schlüsselbereichs wird nicht unterstützt.

    Dies ist ein wichtiger Gesichtspunkt, weil für den Fall, dass die Bereiche einer Tabelle geändert werden müssen, eine neue Tabelle mit den gewünschten Bereichen erstellt und mit den Daten aus der alten Tabelle gefüllt werden muss.
- Doppelte Schlüsselwerte sind nicht zulässig.
- Schlüsselwerte außerhalb des definierten Bereichs sind nicht zulässig.

  Dies gilt nur für Bereichsclustertabellen, die mit DISALLOW OVERFLOW definiert werden.

  – NULL-Werte werden explizit nicht zugelassen.
- Ein Bereichsclusterindex wird nicht physisch gespeichert.

  Ein Index mit RCT-Schlüsselmerkmalen wird in den Systemkatalogen ausgewiesen und kann vom Optimierungsprogramm ausgewählt werden, der Index wird jedoch nicht auf der Platte gespeichert. Bei einer regulären Tabelle wird auch für jeden Index, der einer Tabelle zugeordnet ist, Speicherplatz benötigt. Bei einer Bereichsclustertabelle wird für den RCT-Index kein Speicherbereich benötigt. Das Optimierungsprogramm verwendet die Informationen in den Systemkatalogen, die auf diesen RCT-Index verweisen, um sicherzustellen, dass die korrekte Zugriffsmethode für die Tabelle ausgewählt werden kann.
- Die Erstellung eines Primärschlüssels oder eines eindeutigen Schlüssels in der gleichen Definition wie der des Index für die Bereichsclustertabelle ist nicht zulässig, da sie redundant wäre.
- Bereichsclustertabellen behalten die ursprüngliche Reihenfolge der Schlüsselwerte bei. Durch diese Funktion wird das Clustering der Zeilen innerhalb der Tabelle sichergestellt.

Neben diesen Aspekten gibt es einige Inkompatibilitäten, die entweder die Stellen, an denen Bereichsclustertabellen eingesetzt werden können, oder Dienstprogramme beschränken, die mit diesen Tabellen nicht funktionieren. Folgende Einschränkungen für Bereichsclustertabellen sind zu beachten:

- Deklarierte globale temporäre Tabellen (DGTT) werden nicht unterstützt.

  Für solche temporären Tabellen ist das Bereichsclustermerkmal nicht zulässig.
- Automatisch aktualisierte Übersichtstabellen (AST) werden nicht unterstützt.

  Für solche Tabellen ist das Bereichsclustermerkmal nicht zulässig.
- Das Dienstprogramm LOAD wird nicht unterstützt.

  Zeilen müssen einzeln durch eine Importoperation oder eine Anwendung zum parallelen Einfügen eingefügt werden.
- Das Dienstprogramm REORG TABLE wird nicht unterstützt.

  Bereichsclustertabellen, die mit DISALLOW OVERFLOW werden, brauchen nicht reorganisiert zu werden. Auch für solche Bereichsclustertabellen, die mit ALLOW OVERFLOW definiert sind, ist eine Reorganisation der Daten in diesem Überlaufbereich nicht zulässig.
- Bereichsclustertabellen können sich nur auf einer logischen Maschine befinden.

  Unter Enterprise Server Edition (ESE) mit dem Database Partitioning Feature (DPF) kann sich eine Bereichsclustertabelle nicht in einer Datenbankpartitionsgruppe befinden, die mehr als eine Datenbankpartition enthält. Dies wird ver-

hindert, indem die Erstellung einer Bereichsclustertabelle in einer Datenbankpartitionsgruppe mit mehr als einer Partition nicht zugelassen wird. Darüber hinaus ist auch eine Umverteilung einer Datenbankpartitionsgruppe, die eine Bereichsclustertabelle in einem ihrer Tabellenbereiche enthält, nicht zulässig.

- Der Designadvisor empfiehlt keine Bereichsclustertabellen.
- Bereichsclustertabellen sind per Definition bereits in Clustern angeordnet.

  Dies bedeutet, dass die folgenden Clusteringschemata mit Bereichsclustertabellen nicht kompatibel sind:

  – Tabellen mit mehrdimensionalem Clustering (MDC)
  – Clusterindizes
- Wert- und Standardwertkomprimierung wird nicht unterstützt.
- Umgekehrte Suchoperationen in der Bereichsclustertabelle werden nicht unterstützt.
- Die Option REPLACE im Befehl IMPORT wird nicht unterstützt.
- Die Option WITH EMPTY TABLE in der Anweisung ALTER TABLE ... ACTIVATE NOT LOGGED INITIALLY wird nicht unterstützt.

Zugehörige Konzepte:

- „Bereichsclustertabellen und Satzschlüsselwerte außerhalb des zulässigen Bereichs" auf Seite 157
- „Beispiele für Bereichsclustertabellen" in *Systemverwaltung: Implementierung*

Bereichsclustertabellen und Satzschlüsselwerte außerhalb des zulässigen Bereichs

Die Funktionsweise einer Bereichsclustertabelle (RCT = Range-Clustered Table), in der Überlaufdatensätze erlaubt sind, kann mit Hilfe der Anweisung CREATE TABLE und der Option ALLOW OVERFLOW gesteuert werden. Auf diese Weise können Sie sicherstellen, dass alle für die Tabelle innerhalb des definierten Bereichs erforderlichen Seiten sofort zugeordnet werden können.

Nach der Erstellung arbeiten alle Datensätze mit Schlüsseln, die innerhalb des definierten Bereichs liegen, auf dieselbe Weise. Hierbei spielt es keine Rolle, ob für die Tabelle die Option ALLOW OVERFLOW oder DISALLOW OVERFLOW definiert wurde. Die Unterschiede werden erst dann deutlich, wenn ein Datensatz mit einem Schlüssel auftritt, der außerhalb des definierten Bereichs liegt. Wenn für die Tabelle ALLOW OVERFLOW definiert wurde, wird der betreffende Datensatz dann in den Überlaufbereich gestellt, der dynamisch zugeordnet wird. Wenn weitere Datensätze hinzugefügt werden, die außerhalb des definierten Bereichs liegen, werden diese in den dynamisch vergrößerten Überlaufbereich gestellt. Werden für die Tabelle Aktionen ausgeführt, die diesen Überlaufbereich betreffen, benötigen diese eine längere Verarbeitungszeit, da zur Ausführung der Aktion auf diesen Überlaufbereich zugegriffen werden muss. Je größer der Überlaufbereich ist, desto länger dauert der Zugriff auf diesen Bereich. Nach der ausgedehnten Nutzung des Überlaufbereichs sollten Sie dessen Größe reduzieren, indem Sie die Daten aus der Tabelle in eine neue Bereichsclustertabelle exportieren, die mit neuen erweiterten Bereichen definiert wurde.

In bestimmten Fällen ist es nicht wünschenswert, dass Datensätze in eine Bereichsclustertabelle eingefügt werden und hierbei die Werte der Datensatzschlüssel außerhalb eines zulässigen oder definierten Bereichs liegen. Bei diesem Bereichsclustertabellentyp müssen Sie die Option DISALLOW OVERFLOW der Anweisung

CREATE TABLE verwenden. Nach der Erstellung einer Bereichsclustertabelle dieses Typs werden möglicherweise Fehlernachrichten angezeigt, wenn der Wert eines Datensatzschlüssels außerhalb des zulässigen oder definierten Bereichs liegt.

Zugehörige Referenzen:

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

Sperren bei Bereichsclustertabellen

Im Rahmen der normalen Verarbeitung werden Datensätze gesperrt, um sicherzustellen, dass immer nur eine Anwendung oder ein Benutzer Zugriff auf einen Datensatz bzw. eine Gruppe von Datensätzen hat. Bei Bereichsclustertabellen wird an Stelle der Sperrung von Schlüsseln oder nächsten Schlüsseln die „diskrete Sperrung" verwendet. Bei dieser Methode werden alle Datensätze gesperrt, die von den von der Anwendung bzw. vom Benutzer angeforderten Operationen möglicherweise betroffen sind. Die Anzahl der Sperren, die gesetzt werden müssen, hängt von der gewählten Isolationsstufe ab.

Den Suchbedingungen entsprechende Zeilen in Bereichsclustertabellen, die momentan leer sind, jedoch vorab zugeordnet wurden, werden gesperrt. Hierdurch wird das Sperren des nächsten Schlüssels überflüssig. Zur Erstellung einer Bereichsclustertabelle mit hoher Clusterbildung werden nun weniger Sperren benötigt.

Zugehörige Konzepte:

- „Sperren und Steuerung des gemeinsamen Zugriffs" in *Systemverwaltung: Optimierung*

Tabellen mit mehrdimensionalem Clustering

Das Verfahren des mehrdimensionalen Clustering (MDC - Multidimensional Clustering) stellt eine elegante Methode zur flexiblen, kontinuierlichen und automatischen Anordnung von Daten in Clustern in Bezug auf mehrere Dimensionen bereit. Durch mehrdimensionales Clustering lässt sich die Abfrageleistung erheblich steigern und zudem der Aufwand von Datenpflegeoperationen, wie Reorganisations- und Indexpflegeoperationen bei INSERT-, UPDATE- und DELETE-Operationen bedeutend verringern. Das mehrdimensionale Clustering ist in erster Linie für Data Warehouse- und große Datenbankumgebungen gedacht und kann darüber hinaus in OLTP-Umgebungen (OLTP - Onlinetransaktionsverarbeitung) genutzt werden.

Vergleich zwischen regulären Tabellen und MDC-Tabellen:

Reguläre Tabellen haben datensatzbasierte Indizes. Jedes Clustering der Indizes ist auf eine Dimension beschränkt. Vor Version 8 wurde von DB2® Universal Database (DB2 UDB) lediglich ein eindimensionales Clustering von Daten über Clusterindizes unterstützt. DB2 UDB versucht mit Hilfe eines Clusterindex die physische Reihenfolge von Daten auf den Seiten in der Schlüsselreihenfolge des Index beizubehalten, wenn Datensätze in die Tabelle eingefügt und aktualisiert werden. Clusterindizes verbessern die Leistung von Datenbereichsabfragen immens, die Vergleichselemente haben, die den Schlüssel (oder die Schlüssel) des Clusterindex enthalten. Die Leistung wird durch einen guten Clusterindex erhöht, weil nur auf einen Teil der Tabelle zugegriffen werden muss und ein effizienterer Vorablesezugriff ausgeführt werden kann.

Das Datenclustering über einen Clusterindex hat jedoch auch einige Nachteile. Erstens kann das Clustering nicht gewährleistet werden, da sich der Speicherbereich auf Datenseiten mit der Zeit füllt. Eine Einfügeoperation versucht, einen Datensatz auf einer Seite in der Nähe von solchen Datensätzen einzufügen, die die gleichen oder ähnliche Clusteringschlüsselwerte haben. Wenn jedoch kein Speicherplatz an der Idealposition frei ist, wird der Datensatz an einer anderen Stelle in die Tabelle eingefügt. Daher können zur Wiederherstellung des Clustering der Tabelle sowie zur Einrichtung von Seiten mit zusätzlichem freien Speicher für zukünftige Clustereinfügeanforderungen in regelmäßigen Abständen Tabellenreorganisationen erforderlich sein.

Zweitens kann nur ein Index als „Clusterindex" definiert werden. Alle anderen Indizes sind ohne Clustering, weil die Daten physisch nur in einer Dimension zusammengefasst ('geclustert') werden können. Diese Einschränkung ergibt sich aus der Tatsache, dass der Clusterindex datensatzbasiert ist, so wie dies bei allen Indizes vor Version 8.1 der Fall war.

Drittens können datensatzbasierte Indizes sehr groß sein, weil sie einen Zeiger für jeden einzelnen Datensatz in der Tabelle enthalten.

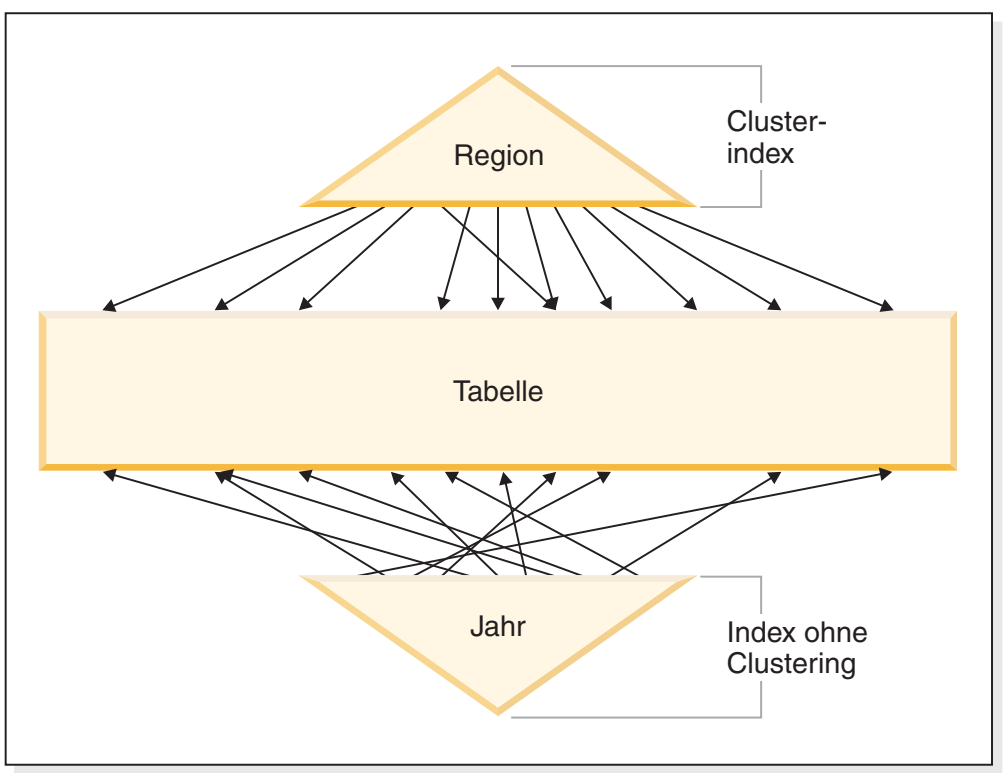


Abbildung 40. Eine reguläre Tabelle mit einem Clusterindex

Für die Tabelle in Abb. 40 sind zwei datensatzbasierte Indizes definiert:

- Ein Clusterindex für „Region"
- Ein anderer Index für „Jahr"

Der Index „Region" ist ein Clusterindex. Dies bedeutet, dass bei der Suche nach Schlüsseln im Index die entsprechenden Datensätze meistenteils auf der gleichen bzw. auf benachbarten Seiten in der Tabelle zu finden sein sollten. Im Unterschied dazu ist der Index „Jahr" ohne Clustering. Das heißt, dass beim Suchen von Schlüsseln in diesem Index die entsprechenden Datensätze wahrscheinlich auf verschiedenen Seiten in der gesamten Tabelle zu finden sind. Suchen über den Clusterindex zeigen eine bessere E/A-Leistung und profitieren umso mehr von sequenziellen Vorablesezugriffen, je mehr die Daten entsprechend diesem Index angeordnet (geclustert) sind.

Mit dem mehrdimensionalen Clustering (MDC) werden Indizes eingeführt, die blockbasiert sind. „Blockindizes" verweisen nicht auf einzelne Datensätze, sondern auf Blöcke bzw. Gruppen von Datensätzen. Durch die physische Anordnung von Daten in einer MDC-Tabelle in Blöcken, die bestimmten Clusteringwerten entsprechen, sowie durch die Verwendung von Blockindizes für den Zugriff auf diese Blöcke kann mehrdimensionales Clustering (MDC) nicht nur alle Nachteile von Clusterindizes wettmachen, sondern bietet darüber hinaus auch erhebliche zusätzliche Leistungsvorteile.

Erstens ermöglicht mehrdimensionales Clustering, die Daten einer Tabelle physisch auf der Grundlage mehrerer Schlüssel bzw. Dimensionen gleichzeitig in Clustern anzuordnen. Mit MDC werden daher die Vorteile eines eindimensionalen Clustering auf mehrere Dimensionen oder Clusteringschlüssel ausgeweitet. Die Abfrageleistung wird überall dort verbessert, wo es ein Clustering einer oder mehrerer angegebener Dimensionen einer Tabelle gibt. Diese Abfragen greifen nicht nur lediglich auf die Seiten zu, die Datensätze mit den richtigen Dimensionswerten enthalten, sondern diese qualifizierten Seiten sind außerdem in Blöcke bzw. EXTENTSIZE große Speicherbereiche gruppiert.

Zweitens kann eine MDC-Tabelle im Gegensatz zu einer Tabelle mit einem Clusterindex, deren Clustering mit der Zeit verloren gehen kann, das Clustering über alle Dimensionen automatisch und kontinuierlich beibehalten und garantieren. Dadurch wird die Notwendigkeit beseitigt, MDC-Tabellen zur Wiederherstellung der physischen Anordnung der Daten zu reorganisieren.

Dritten sind bei MDC die Clusterindizes blockbasiert. Diese Indizes sind beträchtlich kleiner als die regulären datensatzbasierten Indizes, so dass sie wesentlich weniger Plattenspeicherplatz belegen und sich schneller durchsuchen lassen.

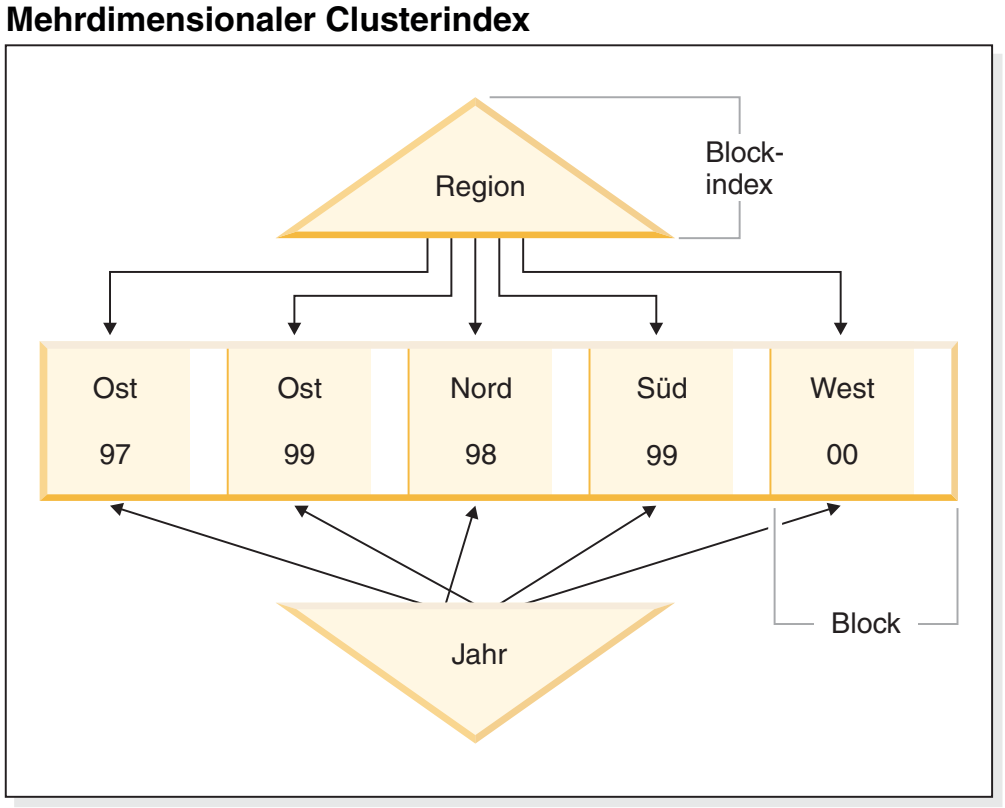


Abbildung 41. Eine Tabelle mit mehrdimensionalem Clustering (MDC)

Blockindizes:

Die MDC-Tabelle in Abb. 41 ist physisch so organisiert, dass Datensätze, die gleiche Werte für „Region" und „Jahr" aufweisen, in getrennte Blöcke bzw. EXTENTSIZE große Speicherbereiche zusammengruppiert werden. Ein Speicherbereich ist eine Gruppe zusammenhängender Seiten auf der Platte, so dass diese Gruppen von Datensätzen in physisch zusammenhängenden Datenseiten zusammengefasst werden. Jede Tabellenseite gehört genau zu einem Block, und alle Blöcke haben die gleiche Größe (d. h. die gleiche Anzahl von Seiten). Die Größe eines Blocks entspricht der Größe eines EXTENTSIZE großen Speicherbereichs des Tabellenbereichs, so dass sich die Blockgrenzen an den Speicherbereichsgrenzen ausrichten. Im oben gezeigten Fall werden zwei Blockindizes erstellt, einer für die Dimension „Region" und ein weiterer für die Dimension „Jahr". Diese Blockindizes enthalten Zeiger, die nur auf die Blöcke in der Tabelle verweisen. Eine Suche im Blockindex „Region" für alle Datensätze, bei denen „Region" gleich „Ost" ist, findet zwei Blöcke, die den Kriterien entsprechen. Alle Datensätze, bei denen „Region" gleich „Ost" ist, und nur diese, werden in diesen beiden Blöcken gefunden, und zwar zusammengefasst in diese beiden Gruppen zusammenhängender Seiten bzw. Speicherbereiche. Völlig unabhängig davon findet eine gleichzeitige Suche nach Datensätzen zwischen 1999 und 2000 im Index „Jahr" drei entsprechende Blöcke. Eine Datensuche in jedem dieser drei Blöcke liefert alle Datensätze, und nur diese Datensätze, die zwischen 1999 und 2000 liegen. Außerdem befinden sich diese Datensätze zusammengefasst auf sequenziell angeordneten Seiten innerhalb der einzelnen Blöcke.

Über diese Clusteringverbesserungen hinaus bieten MDC-Tabellen zudem die folgenden Vorteile:

- Prüfungen und Suchen in Blockindizes laufen aufgrund ihrer im Vergleich zu datensatzbasierten Indizes unglaublich kleinen Größe wesentlich schneller ab.
- Blockindizes und die entsprechende Anordnung von Daten ermöglichen einen sehr differenzierten „Ausschluss von Partitionen" bzw. einen selektiven Tabellenzugriff.
- Abfragen, die Blockindizes nutzen, profitieren von der geringeren Indexgröße, vom optimierten Vorablesezugriff auf Blöcke sowie vom garantierten Clustering der entsprechenden Daten.
- Für einige Abfragen sind weniger Sperren und weniger Aufwand für die Vergleichselementauswertung möglich.
- Blockindizes sind mit wesentlich weniger Protokollier- und Verwaltungsaufwand verbunden, da sie nur aktualisiert werden müssen, wenn der erste Datensatz einem Block hinzugefügt oder der letzte Datensatz aus einem Block gelöscht wird.
- Neu eingefügte Daten können den zusammenhängenden Speicherplatz wiederverwenden, der von Daten freigegeben wurde, die zuvor entfernt wurden.

Anmerkung: Selbst eine MDC-Tabelle, die mit nur einer Dimension definiert wurde, kann von diesen MDC-Attributen profitieren und eine mögliche Alternative zu einer regulären Tabelle mit einem Clusterindex darstellen. Diese Entscheidung sollte auf viele Faktoren gestützt werden, zu denen die Abfragen gehören, die die Auslastung darstellen, sowie die Art und die Verteilung der Daten in der Tabelle. Lesen Sie die Informationen in den Abschnitten zu den Überlegungen bei der Auswahl von Dimensionen und zur MDC-Advisorfunktion im DB2-Advisor.

Wenn Sie eine Tabelle erstellen, können Sie einen oder mehrere Schlüssel als Dimensionen angeben, über die die Daten zu Clustern anzuordnen sind. Jede dieser MDC-Dimensionen kann ähnlich wie bei regulären Indexschlüsseln aus einer oder mehreren Spalten bestehen. Ein *Dimensionsblockindex* wird automatisch für jede angegebene Dimension erstellt und vom Optimierungsprogramm zum schnellen und effizienten Zugriff auf Daten nach der jeweiligen Dimension verwendet. Ein *zusammengesetzter Blockindex* wird ebenfalls automatisch erstellt. Er enthält alle Spalten über alle Dimensionen hinweg und dient zur Beibehaltung des Clustering der Daten bei Einfüge- und Aktualisierungsaktivitäten. Ein zusammengesetzter Blockindex wird nur in dem Fall erstellt, dass eine einzelne Dimension nicht bereits alle Dimensionsschlüsselspalten enthält. Der zusammengesetzte Blockindex kann ebenfalls vom Optimierungsprogramm ausgewählt werden, um effizient auf Daten zuzugreifen, die den Werten einer Untergruppe oder aller der Spaltendimensionen entsprechen.

Anmerkung: Die Zweckmäßigkeit dieses Index bei der Abfrageverarbeitung hängt von der Reihenfolge seiner Schlüsselteile ab. Die Reihenfolge der Schlüsselteile wird durch die Reihenfolge der Spalten bestimmt, die vom Parser bei der Analyse der Dimensionen ermittelt werden, die in der Klausel ORGANIZE BY in der Anweisung CREATE TABLE angegeben wurden. Weitere Informationen finden Sie im Abschnitt „Überlegungen zu Blockindizes für MDC-Tabellen".

Blockindizes sind strukturell mit regulären Indizes identisch, abgesehen von der Tatsache, dass sie nicht auf Datensätze, sondern auf Blöcke zeigen. Blockindizes

sind kleiner als reguläre Indizes, und zwar um einen Faktor der Blockgröße, multipliziert mit der durchschnittlichen Anzahl von Datensätzen auf einer Seite. Die Anzahl von Seiten in einem Block ist gleich dem Wert für EXTENTSIZE des Tabellenbereichs, der zwischen 2 und 256 Seiten betragen kann. Die Seitengröße kann 4, 8, 16 oder 32 KB sein.

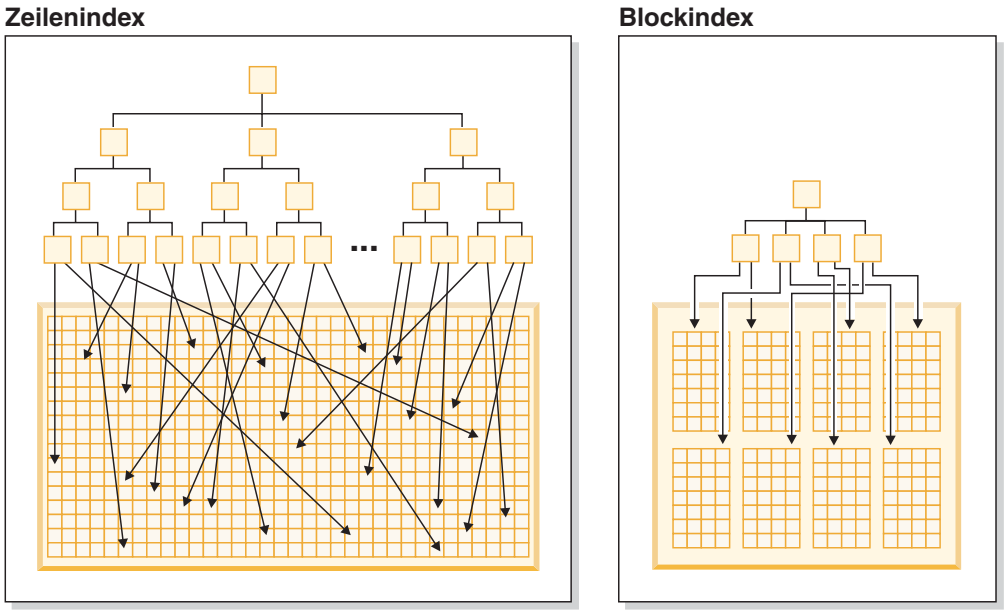


Abbildung 42. Unterschiede zwischen Zeilenindizes und Blockindizes

Wie in Abb. 42 zu sehen ist, gibt es in einem Blockindex jeweils einen Indexeintrag für jeden Block im Unterschied zu einem Eintrag für jede Zeile. Infolgedessen ermöglicht ein Blockindex eine erhebliche Verringerung der Plattenbelegung und einen wesentlich schnelleren Datenzugriff.

In einer MDC-Tabelle bildet jede eindeutige Kombination aus Dimensionswerten eine logische *Zelle*, die sich physisch aus einem oder mehreren Blöcken von Seiten zusammensetzen kann. Der logischen Zelle werden nur so viele Blöcke zugeordnet, wie zur Speicherung der Datensätze erforderlich sind, die den Dimensionswerten dieser logischen Zelle entsprechen. Wenn sich in der Tabelle keine Datensätze befinden, die den Dimensionswerten einer bestimmten logischen Zelle entsprechen, werden dieser logischen Zelle keine Blöcke zugeordnet. Die Gruppe von Blöcken, die Daten mit einem bestimmten Dimensionsschlüsselwert enthalten, werden als *Sektor* (engl. slice) bezeichnet.

Arbeiten mit einer MDC-Tabelle:

Stellen Sie sich zum Beispiel eine MDC-Tabelle mit dem Namen „Umsatz" vor, die Verkaufsdaten für einen landesweit tätigen Einzelhändler aufzeichnet. Das Clustering der Tabelle erfolgt nach den Dimensionen „JahrUndMonat" und „Region". Datensätze in der Tabelle werden in Blöcken gespeichert, die ausreichend aufeinander folgende Seiten enthalten, um einen EXTENTSIZE großen Speicherbereich zu füllen. In Abb. 43 auf Seite 164 wird ein Block durch ein Rechteck dargestellt und nach der logischen Reihenfolge der zugeordneten EXTENTSIZE-Bereiche in der Tabelle durchnummeriert. Die Gitterlinien im Diagramm stellen die logische Einteilung dieser Blöcke dar, wobei jedes Quadrat eine logische Zelle bildet. Eine Spalte oder Zeile im Gitter stellt einen Sektor für eine bestimmte Dimension dar. Zum Beispiel befinden sich alle Datensätze, die in der Spalte „Region" den Wert 'Süd-

Mitte' enthalten, in den Blöcken, die in dem Sektor enthalten sind, der durch die Spalte 'Süd-Mitte' im Gitter definiert wird. Jeder Block in diesem Sektor enthält ebenfalls nur Datensätze, die im Feld „Region" den Wert 'Süd-Mitte' aufweisen. Das heißt, ein Block ist nur dann in diesem Sektor oder dieser Spalte des Gitters enthalten, wenn er Datensätze enthält, die den Wert 'Süd-Mitte' im Feld „Region" besitzen.

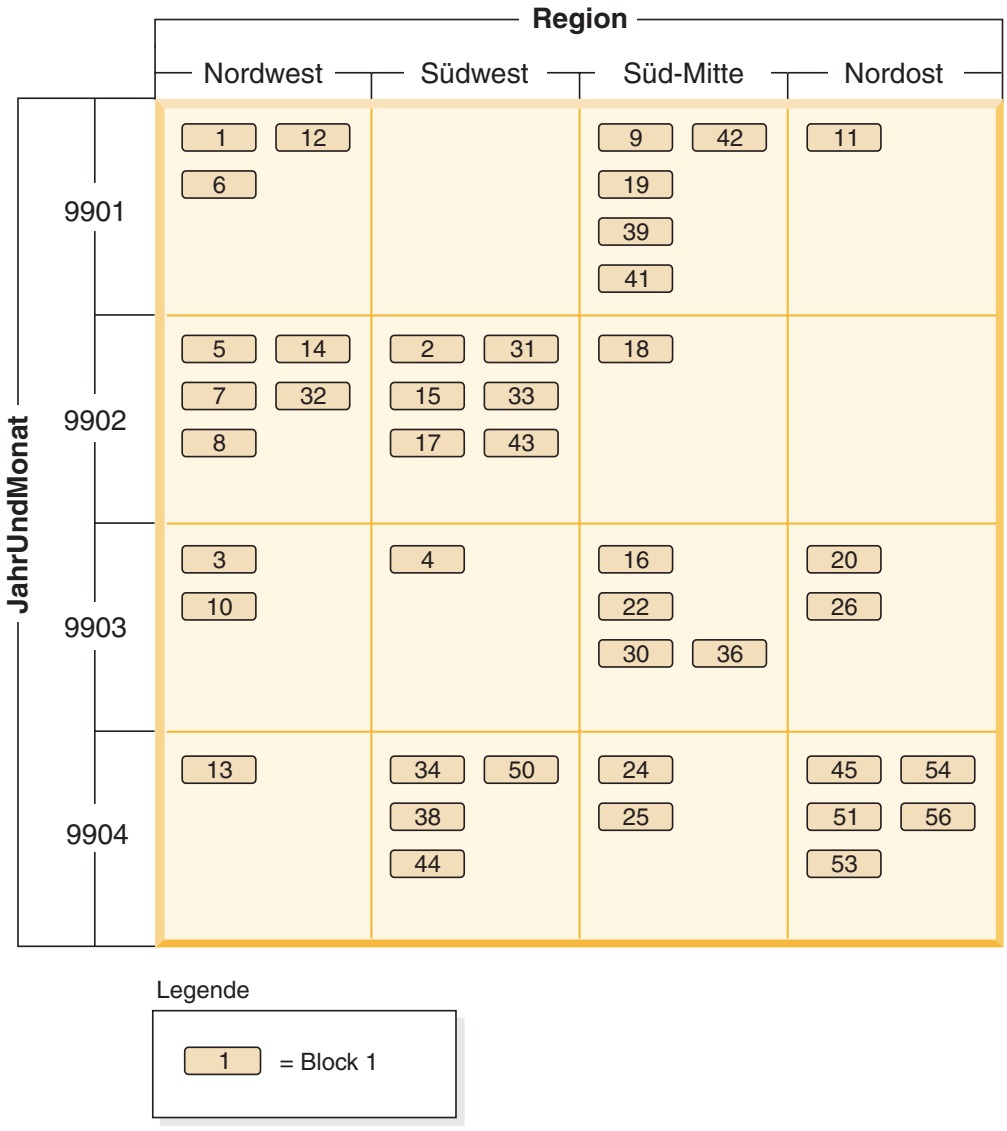


Abbildung 43. Mehrdimensionale Tabelle mit den Dimensionen 'Region' und 'JahrUndMonat' mit dem Namen 'Umsatz'

Zur Bestimmung, welche Blöcke einen Sektor bilden oder, damit äquivalent, welche Blöcke alle Datensätze mit einem bestimmten Dimensionsschlüsselwert enthalten, wird bei der Erstellung der Tabelle automatisch ein Dimensionsblockindex für jede Dimension erstellt.

In Abb. 44 auf Seite 165 wurde ein Dimensionsblockindex für die Dimension „JahrUndMonat" und ein weiterer für die Dimension „Region" erstellt. Jeder Dimensionsblockindex ist in gleicher Weise wie ein herkömmlicher Satz-ID-Index (RID-Index) strukturiert, abgesehen davon, dass die Schlüssel auf der Blattebene auf eine Block-ID (BID) und nicht auf eine Satz-ID (RID) verweisen. Eine Satz-ID

(RID) gibt die Position eines Datensatzes in der Tabelle durch eine physische Seitennummer und eine Positionsnummer an, das heißt, der Position auf der Seite, an der sich der Datensatz befindet. Eine Block-ID (BID) stellt einen Block durch die physische Seitennummer der ersten Seite des entsprechenden EXTENTSIZE großen Speicherbereichs und einer Dummy-Position (0) dar. Da alle Seiten im Block angefangen bei dieser Seite physisch aufeinander folgen und die Größe des Blocks bekannt ist, können mit Hilfe dieser BID alle Datensätze im Block gefunden werden.

Ein Sektor bzw. die Menge von Blöcken, die Seiten mit allen Datensätzen enthalten, die einen bestimmten Schlüsselwert in einer Dimension haben, werden im zugeordneten Dimensionsblockindex durch eine BID-Liste für diesen Schlüsselwert dargestellt.

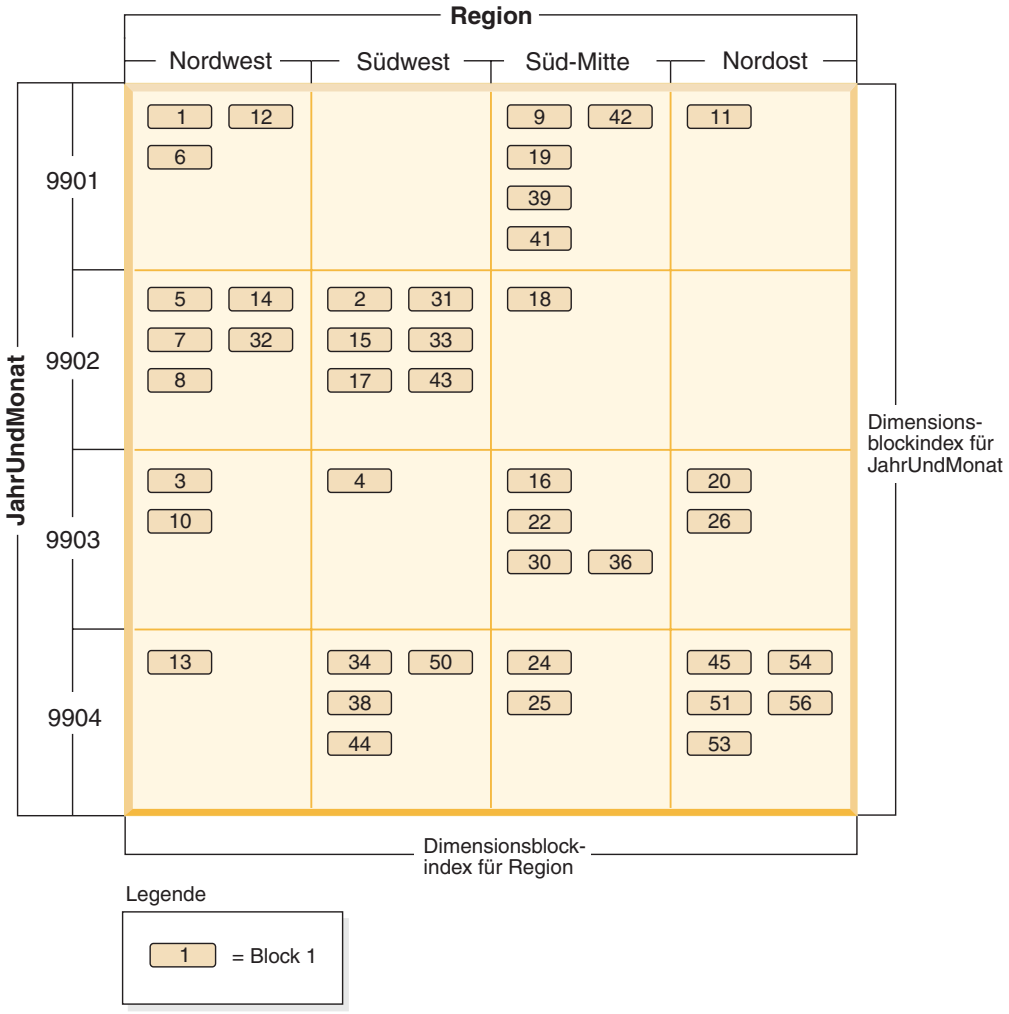


Abbildung 44. Tabelle 'Umsatz' mit den Dimensionen 'Region' und 'JahrUndMonat' mit gezeigten Dimensionsblockindizes

Abb. 45 auf Seite 166 zeigt, wie ein Schlüssel aus dem Dimensionsblockindex für „Region" aussehen würde. Der Schlüssel besteht aus einem Schlüsselwert, in diesem Fall 'Süd-Mitte', sowie einer Liste von Block-IDs (BIDs). Jede BID enthält eine Blockposition. In Abb. 45 auf Seite 166 stimmen die aufgeführten Blocknummern

mit denen überein, die sich im Sektor 'Süd-Mitte' im Gitter der Umsatztabelle befinden (siehe Abb. 43 auf Seite 164).

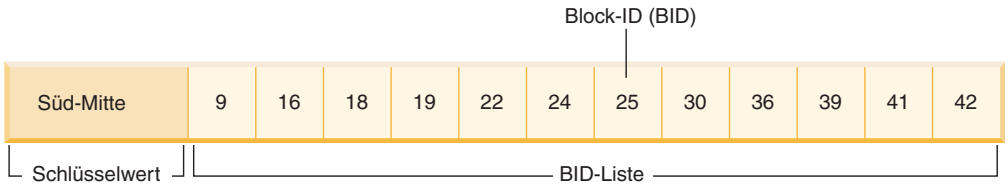


Abbildung 45. Schlüssel aus dem Dimensionsblockindex für 'Region'

Analog lässt sich die Liste von Blöcken, die alle Datensätze mit dem Wert '9902' für die Dimension „JahrUndMonat" enthalten, im Dimensionsblockindex für „JahrUndMonat" finden, der in Abb. 46 gezeigt wird.

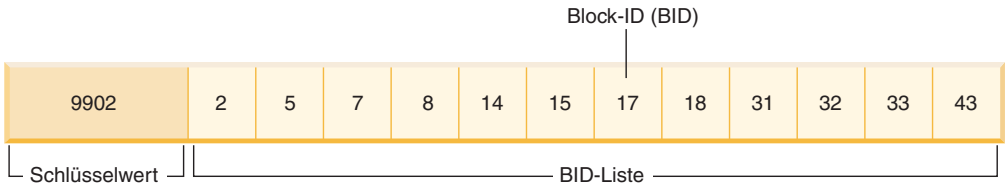


Abbildung 46. Schlüssel aus dem Dimensionsblockindex für 'JahrUndMonat'

Blockindizes und Abfrageleistung:

Suchen in einem der Blockindizes einer MDC-Tabelle ermöglichen einen Clusterdatenzugriff, da jede BID einer Gruppe sequenzieller Seiten in der Tabelle entspricht, die garantiert Daten enthalten, die den angegebenen Dimensionswert haben. Darüber hinaus kann auf Dimensionen oder Sektoren unabhängig voneinander über die zugehörigen Blockindizes zugegriffen werden, ohne den Clusterfaktor einer anderen Dimension oder eines anderen Sektors zu beeinträchtigen. Dies ermöglicht eine Mehrdimensionalität des mehrdimensionalen Clustering.

Abfragen, die die Vorteile des Blockindexzugriffs nutzen, können von einer Reihe Faktoren profitieren, die sich leistungssteigernd auswirken. Erstens ist der Blockindex wesentlich kleiner als ein regulärer Index und bietet so eine hohe Sucheffizienz. Zweitens hängt der Vorablesezugriff auf die Datenseiten nicht von der Sequenzerkennung ab, wenn Blockindizes verwendet werden. DB2 UDB durchsucht den Index vorausschauend und liest die Datenseiten von Blöcken in den Speicher durch E/A-Operationen in großen Blöcken ein, so dass sichergestellt wird, dass die Suche keinen E/A-Aufwand verursacht, wenn auf die Datenseiten in der Tabelle zugegriffen wird. Drittens werden die Daten in der Tabelle auf sequenziell aufeinander folgenden Seiten zusammengefasst, wodurch die Ein-/Ausgabe optimiert und die Ergebnismenge nur aus einem ausgewählten Abschnitt der Tabelle ermittelt wird. Viertens werden bei Verwendung eines blockbasierten Pufferpools mit einer Blockgröße, die dem Wert für EXTENTSIZE entspricht, MDC-Blöcke aus sequenziell angeordneten Seiten auf der Platte in sequenziell angeordnete Seiten im Arbeitsspeicher vorab eingelesen, wodurch sich der Effekt des Clustering auf die Leistung noch weiter erhöht. Und schließlich werden die Datensätze aus den einzelnen Blöcken mit Hilfe einer relationalen Minisuche in den zugehörigen Datenseiten abgerufen, was in vielen Fällen eine schnellere Datensuchmethode ist als ein RID-basierter Abruf.

Abfragen können Blockindizes zur Eingrenzung der Suche auf einen Abschnitt der Tabelle verwenden, der einen bestimmten Dimensionswert oder -wertebereich enthält. Dies ermöglicht eine differenzierte Form eines „Partitionsausschlusses", das heißt, eines Blockausschlusses. Infolgedessen kann sich für die Tabelle ein besserer gleichzeitiger Zugriff ergeben, weil andere Abfragen, Ladeoperationen, Einfügungen, Aktualisierungen und Löschungen vielleicht auf andere Blöcke in der Tabelle zugreifen, ohne mit der Datenmenge dieser einen Abfrage in Berührung zu kommen.

Wenn die Tabelle 'Umsatz' nach drei Dimensionen in Clustern angeordnet ist, können auch die einzelnen Dimensionsblockindizes zum Auffinden einer Gruppe von Blöcken verwendet werden, die Datensätze enthalten, die eine Abfrage auf eine Untergruppe aller Dimensionen der Tabelle erfüllen. Wenn die Tabelle die Dimensionen „JahrUndMonat", „Region" und „Produkt" besitzt, kann diese Tabelle als logischer Würfel gedacht werden, wie in Abb. 47 veranschaulicht.

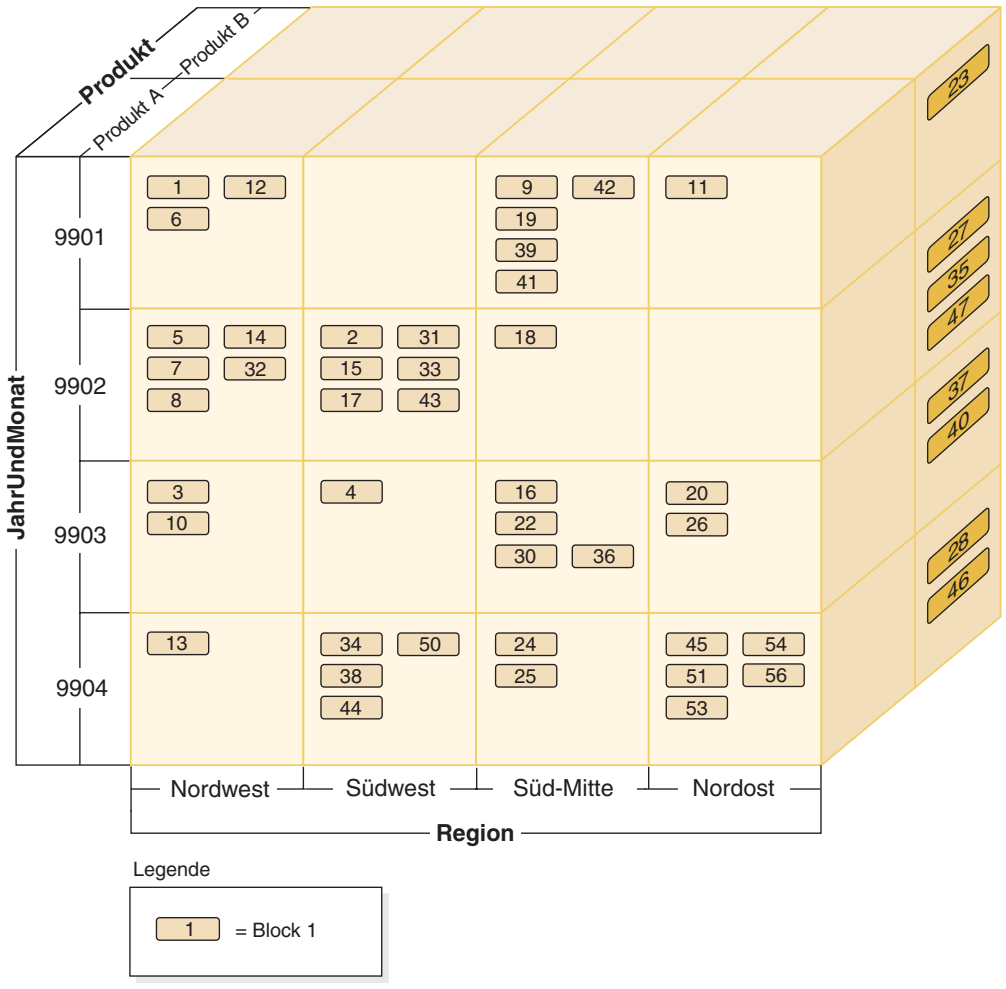


Abbildung 47. Mehrdimensionale Tabelle mit den Dimensionen 'Region', 'JahrUndMonat' und 'Produkt'

Für die MDC-Tabelle in Abb. 47 auf Seite 167 werden vier Blockindizes erstellt: je einer für die einzelnen Dimensionen „JahrUndMonat", „Region" und „Produkt" sowie ein weiterer mit einem Schlüssel, der all diese Dimensionsspalten umfasst. Um alle Datensätze abzurufen, bei denen „Produkt" gleich „Produkt A" und „Region" gleich „Nordost" gilt, würde DB2 UDB zunächst nach dem Schlüssel für Produkt A im Dimensionsblockindex für „Produkt" suchen. (Siehe Abb. 48.) Anschließend bestimmt DB2 UDB durch Suchen des Schlüssels „Nordost" im Dimensionsblockindex „Region" die Blöcke, die alle Datensätze mit „Region" gleich „Nordost" enthalten. (Siehe Abb. 49.)

Produkt A	1	2	3	...	11	...	20	22	24	25	26	30	...	56

Abbildung 48. Schlüssel aus dem Dimensionsblockindex für 'Produkt'

Nordost	11	20	23	26	27	28	35	37	40	45	46	47	51	53	54	56

Abbildung 49. Schlüssel aus dem Dimensionsblockindex für 'Region'

Blockindexsuchen können durch die logischen Operatoren UND (AND) und ODER (OR) kombiniert werden, wobei die zu durchsuchende resultierende Blockliste ebenfalls einen Clusterdatenzugriff ermöglicht.

Um im obigen Beispiel die Menge von Blöcken zu finden, die alle Datensätze mit beiden Dimensionswerten enthalten, müssen Sie die Schnittmenge dieser beiden Sektoren ermitteln. Dies geschieht durch eine logische UND-Operation zwischen den Block-ID-Listen aus den beiden Blockindexschlüsseln. Die gemeinsamen BID-Werte sind 11, 20, 26, 45, 54, 51, 53 und 56.

Das folgende Beispiel veranschaulicht, wie mit Hilfe der logischen ODER-Operation für Blockindizes eine Abfrage mit Vergleichselementen erfüllt wird, die sich auf zwei Dimensionen beziehen. In Abb. 50 auf Seite 169 wird eine MDC-Tabelle angenommen, die die beiden Dimensionen „Farbe" und „Land" besitzt. Das Ziel besteht darin, alle Datensätze in der MDC-Tabelle abzurufen, auf welche die Bedingungen „Farbe" gleich „blau" oder „Land" gleich „USA" zutreffen.

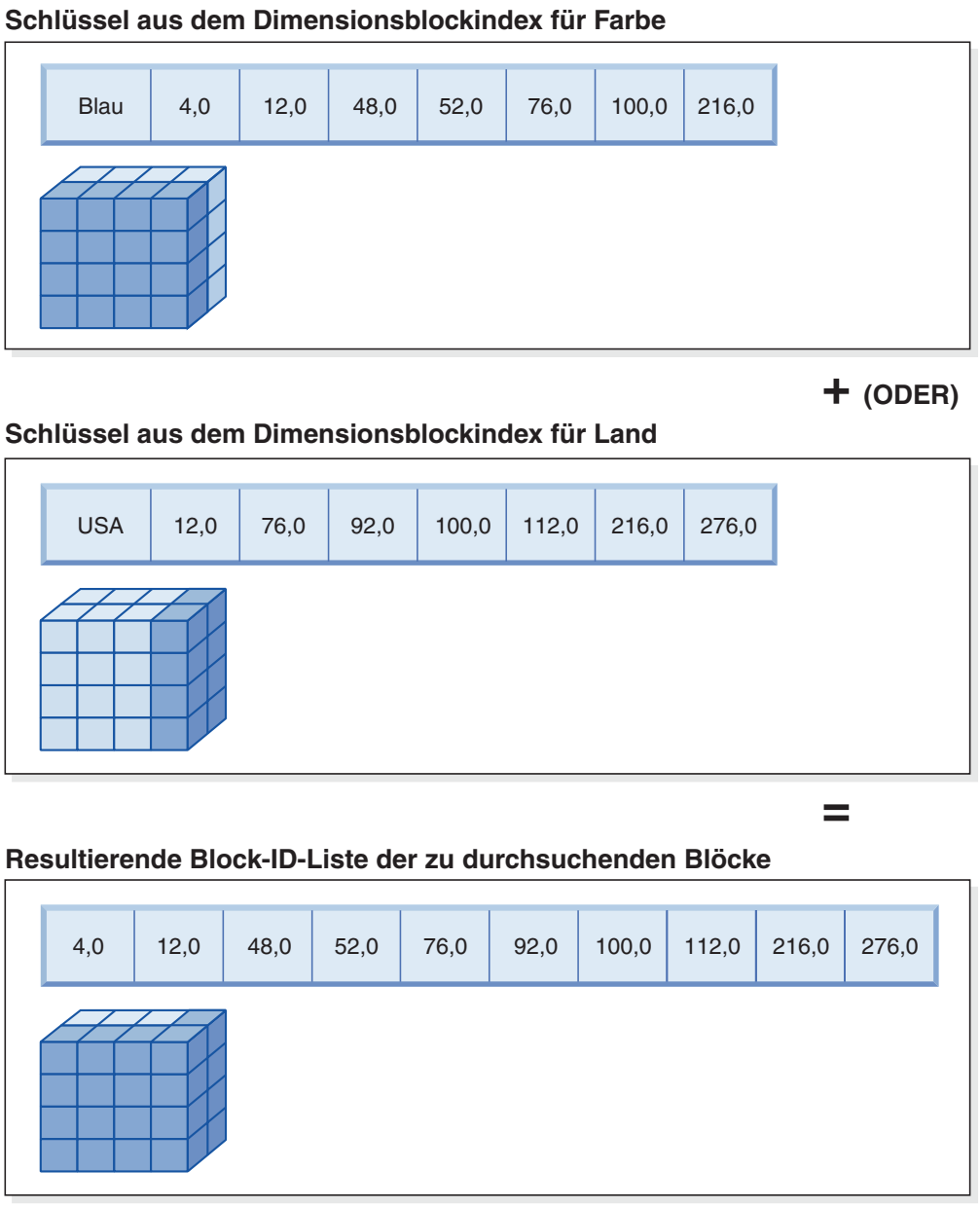


Abbildung 50. Verwendung der logischen ODER-Operation für Blockindizes

Diese Abbildung zeigt, wie die Ergebnisse aus zwei getrennten Blockindexsuchen zur Bestimmung des Bereichs von Werten, die den Einschränkungen der Vergleichselemente entsprechen, kombiniert werden können.

Auf der Grundlage der Vergleichselemente aus der SELECT-Anweisung werden zwei getrennte Suchen in den Dimensionsblockindizes ausgeführt: eine nach dem Sektor 'Blau' und eine nach dem Sektor 'USA'. Im Arbeitsspeicher wird eine logische ODER-Operation ausgeführt, um die Vereinigungsmenge der beiden Sektoren zu ermitteln und die kombinierte Gruppe von Blöcken, die sich in beiden Sektoren befinden, zu bestimmen (was eine Eliminierung doppelter Blöcke mit einschließt).

Wenn eine Liste von Blöcken zum Durchsuchen ermittelt ist, kann DB2 UDB eine relationale Minisuche in den einzelnen Blöcken durchführen. Dabei kann ein Vorablesezugriff auf die Blöcke stattfinden, der gerade eine E/A-Operation pro Block erforderlich macht, da jeder Block als ein EXTENTSIZE großer Speicherbereich auf dem Datenträger gespeichert ist und als Einheit in den Pufferpool eingelesen werden kann. Wenn Vergleichselemente auf die Daten angewendet werden müssen, müssen Dimensionsvergleichselemente nur auf einen Datensatz im Block angewendet werden, weil alle Datensätze im Block garantiert die gleichen Dimensionsschlüsselwerte besitzen. Wenn weitere Vergleichselemente vorhanden sind, muss DB2 UDB diese nur an den übrigen Datensätzen im Block überprüfen.

MDC-Tabellen unterstützen außerdem reguläre RID-basierte Indizes. RID- und Blockindizes können durch eine logische UND-Operation oder eine ODER-Operation kombiniert werden. Blockindizes stellen dem Optimierungsprogramm zusätzliche Zugriffspläne zur Auswahl, verhindern andererseits die Verwendung der traditionellen Zugriffspläne (z. B. RID-Suchen, Verknüpfungen und Tabellensuchen) nicht. Blockindexpläne werden durch das Optimierungsprogramm wie alle anderen möglichen Zugriffspläne für eine bestimmte Abfrage hinsichtlich ihres Aufwands geprüft, und der günstigste Plan wird ausgewählt.

Der DB2-Designadvisor kann RID-basierte Indizes für MDC-Tabellen oder MDC-Dimensionen für eine Tabelle empfehlen.

Automatisches Beibehalten des Clustering bei INSERT-Operationen:

Die automatische Beibehaltung des Datenclustering in einer MDC-Tabelle wird durch den zusammengesetzten Blockindex sichergestellt. Er wird zur dynamischen Verwaltung und Beibehaltung des physischen Clustering (Gruppierung) von Daten in den Dimensionen der Tabelle über die Folge von INSERT-Operationen hinweg verwendet. In diesem zusammengesetzten Blockindex findet sich nur je ein Schlüssel für diejenigen logischen Zellen der Tabelle, die Datensätze enthalten. Dieser Blockindex dient bei einer INSERT-Operation so zur raschen und effizienten Feststellung, ob eine logische Zelle in der Tabelle vorhanden ist. Nur wenn dies der Fall ist, wird außerdem festgestellt, welche Blöcke Datensätze mit der bestimmten Gruppe von Dimensionswerten dieser Zelle enthalten.

Wenn eine INSERT-Operation ausgeführt wird, geschieht Folgendes:

- Der zusammengesetzte Blockindex wird auf das Vorhandensein der logischen Zelle sondiert, die den Dimensionswerten des einzufügenden Datensatzes entspricht.
- Wenn der Schlüssel der logischen Zelle im Index gefunden wird, liefert die zugehörige Block-ID-Liste (BIDs) eine vollständige Liste der Blöcke in der Tabelle, die die Dimensionswerte der logischen Zelle enthalten. (Siehe Abb. 51 auf Seite 171.) Dies begrenzt die Anzahl von EXTENTSIZE großen Speicherbereichen der Tabelle, die nach Platz zum Einfügen des Datensatzes zu durchsuchen sind.
- Wenn der Schlüssel der logischen Zelle im Index nicht gefunden wird oder wenn die Speicherbereiche, die diese Werte enthalten, voll sind, wird der logischen Zelle ein neuer Block zugeordnet. Wenn möglich, erfolgt zuerst eine Wiederverwendung eines leeren Blocks in der Tabelle, bevor die Tabelle um einen neuen, EXTENTSIZE großen Speicherbereich von Seiten (einen neuen Block) erweitert wird.

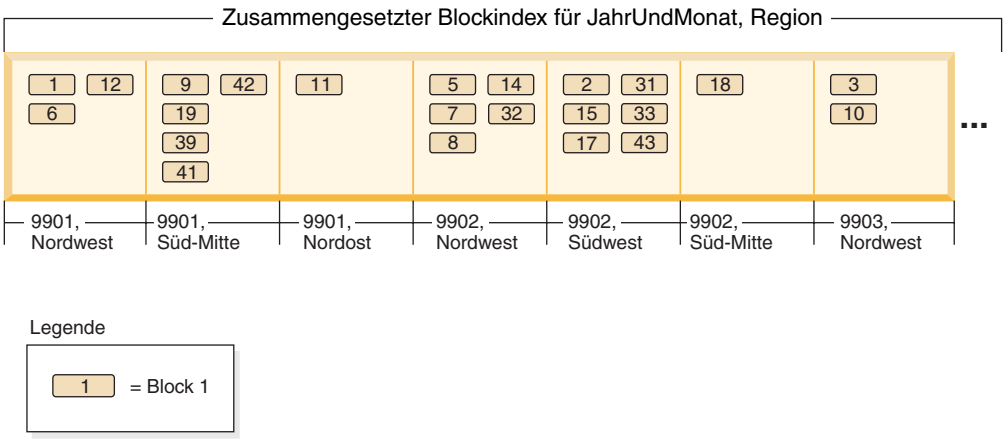


Abbildung 51. Zusammengesetzter Blockindex für 'JahrUndMonat', 'Region'

Wie bereits erläutert, werden Datensätze mit bestimmten Dimensionswerten garantiert in einer Gruppe von Blöcken gefunden, die ausschließlich und gleichzeitig sämtliche Datensätze mit diesen Werten enthalten. Blöcke bestehen aus aufeinander folgenden Seiten auf der Platte. Infolgedessen erfolgt der Zugriff auf diese Datensätze sequenziell und ermöglicht ein Clustering. Dieses Clustering wird über die Zeit automatisch beibehalten, indem sichergestellt wird, dass Datensätze nur in Blöcke aus Zellen mit den Dimensionswerten der jeweiligen Datensätze eingefügt werden. Wenn vorhandene Blöcke in einer logischen Zelle voll sind, wird entweder ein leerer Block wiederverwendet oder ein neuer Block zugeordnet und der Gruppe von Blöcken für diese logische Zelle hinzugefügt. Wenn ein Block seiner Datensätze entleert wird, wird die Block-ID (BID) aus den Blockindizes entfernt. Dies hebt die Zuordnung des Blocks zu den Werten einer logischen Zelle auf, so dass er in Zukunft von einer anderen logischen Zelle wiederverwendet werden kann. Auf diese Weise werden Zellen und die ihnen zugeordneten Blockindexeinträge der Tabelle dynamisch hinzugefügt und aus ihr entfernt, um je nach Bedarf nur die Daten unterzubringen, die in der Tabelle vorhanden sind. Der zusammengesetzte Blockindex dient zur Verwaltung dieser Funktion, da er logische Zellwerte den Blöcken zuordnet, die Datensätze mit diesen Werten enthalten.

Da das Clustering auf diese Weise automatisch beibehalten wird, wird eine Reorganisation einer MDC-Tabelle zu keinem Zeitpunkt erforderlich, um das Clustering der Daten wiederherzustellen. Allerdings kann eine Reorganisation weiterhin zur Wiedergewinnung von Speicherplatz durchgeführt werden. Wenn Zellen zum Beispiel zahlreiche dünn gefüllte Blöcke haben, deren Daten in weniger Blöcke passen würden, oder wenn die Tabelle viele Zeiger-Überlauf-Paare hat, würde eine Reorganisation der Tabelle die Datensätze, die zu den einzelnen logischen Zellen gehören, in die kleinstmögliche Anzahl erforderlicher Blöcke zusammenfassen und die Zeiger-Überlauf-Paare entfernen.

Das folgende Beispiel veranschaulicht, wie der zusammengesetzte Blockindex zur Abfrageverarbeitung verwendet werden kann. Wenn Sie alle Datensätze der Tabelle 'Umsatz' abrufen wollen, die den Wert 'Nordwest' für „Region" und den Wert '9903' für „JahrUndMonat" haben, würde DB2 UDB den Schlüsselwert '9903, Nordwest' im zusammengesetzten Blockindex aufsuchen, wie in Abb. 52 auf Seite 172 gezeigt. Der Schlüssel besteht aus einem Schlüsselwert, in diesem Fall '9903, Nordwest', und einer Liste von BIDs. Wie Sie sehen, sind nur die BIDs 3 und 10 aufgelistet. Und tatsächlich sind in der Tabelle 'Umsatz' nur zwei Blöcke vorhanden, die Datensätze mit diesen beiden Werten enthalten.

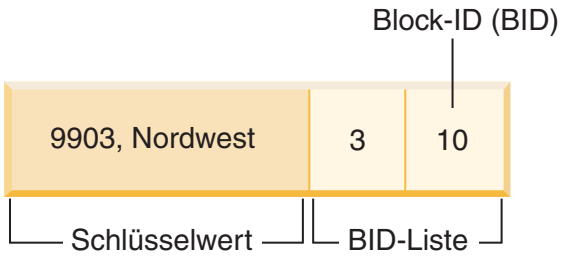


Abbildung 52. Schlüssel aus dem zusammengesetzten Blockindex für 'JahrUndMonat', 'Region'

Zur Veranschaulichung, wie der zusammengesetzte Blockindex bei Einfügeoperationen verwendet wird, soll das Beispiel der Einfügung eines weiteren Datensatzes mit den Dimensionswerten '9903' und 'Nordwest' betrachtet werden. DB2 UDB würde diesen Schlüsselwert im zusammengesetzten Blockindex suchen und die BIDs für die Blöcke 3 und 10 finden. Diese Blöcke enthalten alle Datensätze und nur die Datensätze, die diese Dimensionsschlüsselwerte besitzen. Wenn Speicherplatz verfügbar ist, fügt DB2 UDB den neuen Datensatz in einen dieser Blöcke ein. Wenn kein Platz in den Seiten dieser Blöcke mehr vorhanden ist, ordnet DB2 UDB einen neuen Block für die Tabelle zu oder verwendet einen zuvor geleerten Block in der Tabelle. Beachten Sie, dass in diesem Beispiel Block 48 zurzeit nicht von der Tabelle verwendet wird. DB2 UDB fügt den Datensatz in den Block ein und ordnet diesen Block der aktuellen logischen Zelle zu, indem DB2 UDB die Block-ID (BID) des Blocks dem zusammengesetzten Blockindex sowie den einzelnen Dimensionsblockindizes hinzufügt. Abb. 53 stellt die Schlüssel der Dimensionsblockindizes nach dem Hinzufügen von Block 48 dar.

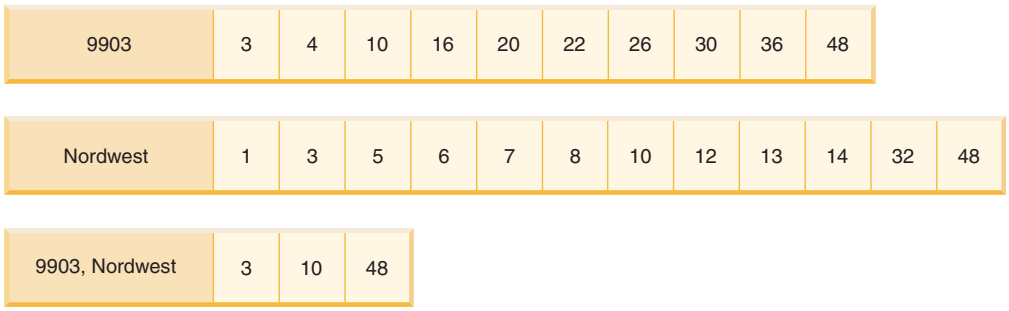


Abbildung 53. Schlüssel aus den Dimensionsblockindizes nach Hinzufügen von Block 48

Blockzuordnung:

Wenn ein Block geleert wird, wird seine Zuordnung zu den Werten der aktuellen logischen Zelle aufgehoben, indem seine Block-ID aus den Blockindizes entfernt wird. Der Block kann anschließend von einer anderen logischen Zelle wiederverwendet werden. Dies verringert den Bedarf an neuen Blöcken zur Erweiterung der Tabelle. Wenn ein neuer Block benötigt wird, müssen die zuvor geleerten Blöcke rasch gefunden werden können, ohne dass die Tabelle nach ihnen durchsucht werden muss.

Die Blockzuordnung (engl. block map) ist eine neue Struktur, die eine Lokalisierung leerer Blöcke in der MDC-Tabelle vereinfacht. Die Blockzuordnung wird als separates Objekt gespeichert:

- In SMS-Tabellenbereichen als separate .BKM-Datei
- In DMS-Tabellenbereichen als neuer Objektdeskriptor in der Objekttabelle

Die Blockzuordnung ist eine Feldgruppe (Array), die für jeden Block der Tabelle einen Eintrag enthält. Jeder Eintrag besteht aus einem Satz von Statusbit für einen Block. Die Statusbit geben unter anderem folgende Status an:

- In Gebrauch. Der Block ist einer logischen Zelle zugeordnet.
- Laden. Der Block wurde kürzlich geladen und ist für Suchoperationen noch nicht sichtbar.
- Integritätsprüfung. Der Block wurde kürzlich geladen, die Integritätsprüfung steht noch aus.
- Aktualisieren. Der Block wurde kürzlich geladen, gespeicherte Abfragesichten müssen noch aktualisiert werden.

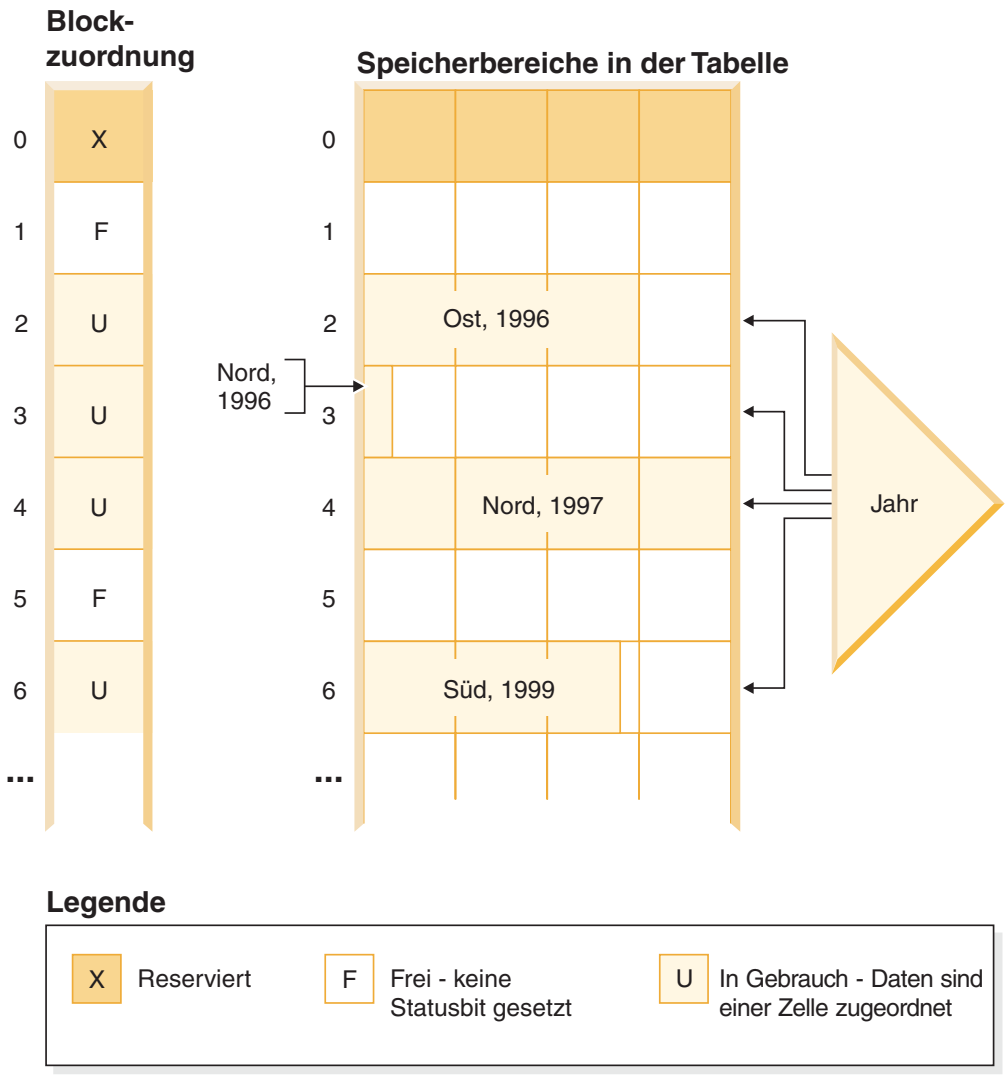


Abbildung 54. Funktionsweise einer Blockzuordnung

Die linke Seite der Abbildung zeigt die Blockzuordnungsfeldgruppe mit verschiedenen Einträgen für die Blöcke in der Tabelle. Die rechte Seite der Abbildung zeigt, wie die einzelnen EXTENTSIZE großen Speicherbereiche der Tabelle verwendet werden: einige sind frei, die meisten in Gebrauch. Datensätze werden nur in den Blöcken gefunden, die in der Blockzuordnung als frei markiert sind. Aus Gründen der Übersichtlichkeit zeigt die Abbildung nur einen der beiden Dimensionsblockindizes.

Anmerkungen:

1. Im Blockindex gibt es nur Zeiger auf Blöcke, die in der Blockzuordnung als „In Gebrauch" markiert sind.
2. Der erste Block ist reserviert. Dieser Block enthält Systemdatensätze für die Tabelle.

Freie Blöcke zur Verwendung in einer Zelle lassen sich durch eine Suche nach als „frei" markierten Blöcken (d. h. Blöcke, für die keine Bit gesetzt sind) in der Blockzuordnung leicht finden.

Tabellensuchen verwenden die Blockzuordnung ebenfalls, um nur auf Speicherbereiche zuzugreifen, die momentan Daten enthalten. Alle Speicherbereiche, die nicht in Gebrauch sind, brauchen in der Tabellensuche nicht berücksichtigt zu werden. Zur Veranschaulichung: Eine Tabellensuche würde in diesem Beispiel (Abb. 54 auf Seite 173) beim dritten Speicherbereich (Speicherbereich 2) in der Tabelle beginnen, indem sie den ersten reservierten Speicherbereich und den darauf folgenden leeren Speicherbereich überspringt, die Blöcke 2, 3 und 4 in der Tabelle durchsuchen, den nächsten Speicherbereich überspringen (ohne eine der Datenseiten dieses Speicherbereichs zu berühren) und schließlich die Suche von dort fortsetzen.

Löschoperationen in MDC-Tabellen:

Wenn ein Datensatz in einer MDC-Tabelle gelöscht wird, löscht DB2 UDB lediglich diesen Datensatz, sofern es sich nicht um den letzten Datensatz in einem Block handelt, und entfernt die Satz-ID (RID) aus allen datensatzbasierten Indizes, die für die Tabelle definiert sind. Wenn eine Löschoperation jedoch den letzten Datensatz aus einem Block entfernt, gibt DB2 UDB den Block frei, indem das Statusbit „In Gebrauch" (IN_USE) geändert und die Block-ID (BID) aus allen Blockindizes entfernt werden. Falls datensatzbasierte Indizes vorhanden sind, wird die entsprechende Satz-ID natürlich auch aus diesen entfernt.

Anmerkung: Das heißt, Blockindexeinträge müssen nur einmal für den ganzen Block entfernt werden, und dies nur, wenn der Block vollständig geleert wird, im Gegensatz zu Einträgen in datensatzbasierten Indizes, die jeweils für die gelöschte Zeile entfernt werden müssen.

Aktualisierungen in MDC-Tabellen:

In einer MDC-Tabelle werden Werte, die nicht zu einer Dimension gehören, ebenso wie bei regulären Tabellen an Ort und Stelle vorgenommen. Wenn sich die Aktualisierung auf eine Spalte mit variabler Länge auswirkt und der Datensatz nicht mehr auf die Seite passt, wird eine andere Seite mit ausreichend Speicherplatz gefunden. Die Suche nach dieser neuen Seite beginnt im selben Block. Wenn in diesem Block kein geeigneter Speicherplatz vorhanden ist, wird der Algorithmus zum Einfügen eines neuen Datensatzes verwendet, um eine Seite in der logischen Zelle mit genügend Speicherplatz zu finden. Die Blockindizes müssen nur in dem Fall aktualisiert werden, dass kein Speicherplatz in der Zelle gefunden wird und der Zelle ein neuer Block hinzugefügt werden muss.

Aktualisierungen von Dimensionswerten werden als Löschung des aktuellen Datensatzes mit anschließender Einfügung des geänderten Datensatzes behandelt, weil der Datensatz die logische Zelle wechselt, zu der er gehört. Wenn die Löschung des aktuellen Datensatzes dazu führt, dass ein Block vollständig

geleert wird, muss der Blockindex aktualisiert werden. Analog muss der Blockindex aktualisiert werden, wenn die Einfügung des neuen Datensatzes einen neuen Block erforderlich macht.

Blockindizes müssen nur aktualisiert werden, wenn der erste Datensatz in einen Block eingefügt bzw. der letzte Datensatz aus einem Block gelöscht wird. Der Aufwand der Verwaltung und Protokollierung für Blockindizes ist daher wesentlich geringer als der Aufwand, der mit regulären Indizes verbunden ist. Für jeden Blockindex, der ansonsten ein regulärer Index gewesen wäre, ist der Verwaltungs- und Protokollaufwand beträchtlich reduziert.

MDC-Tabellen werden wie jede andere vorhandene Tabelle behandelt. Das bedeutet, dass für sie in gleicher Weise Auslöser, referenzielle Integritätsbedingungen, Sichten und gespeicherte Abfragetabellen definiert werden können.

Überlegungen zum Laden von Daten in MDC-Tabellen:

Wenn Sie regelmäßig Daten in Ihr Data Warehouse versetzen, können Sie MDC-Tabellen vorteilhaft nutzen. In MDC-Tabellen verwendet das Programm LOAD zunächst geleerte Blöcke in der Tabelle wieder, bevor es die Tabelle erweitert und neue Blöcke für die verbleibenden Daten hinzufügt. Wenn Sie eine Datengruppe gelöscht haben, zum Beispiel alle alten Daten für einen Monat, können Sie das Dienstprogramm LOAD verwenden, um die Daten des nächsten Monats einzuladen, wobei die Blöcke wiederverwendet werden können, die geleert wurden, nachdem die Löschoperation festgeschrieben wurde.

Beim Laden von Daten in MDC-Tabellen können die Eingabedaten sortiert oder unsortiert sein. Wenn die Daten unsortiert sind, sind folgende Maßnahmen zu beachten:

- Erhöhen Sie den Wert des Konfigurationsparameters *util_heap*.

  Eine Erhöhung der Zwischenspeichergröße für Dienstprogramme wirkt sich auf alle Ladeoperationen in der Datenbank (sowie auf Sicherungs- und Wiederherstellungsoperationen) aus.
- Erhöhen Sie den Wert, der mit der Klausel DATA BUFFER des Befehls LOAD angegeben wird.

  Eine Erhöhung dieses Werts wirkt sich nur auf eine Ladeanforderung aus. Die Zwischenspeichergröße für Dienstprogramme muss ausreichen, um möglicherweise mehrere gleichzeitig zu verarbeitende Ladeanforderungen unterzubringen.
- Stellen Sie sicher, dass die für den Pufferpool verwendete Seitengröße mit der größten Seitengröße für den temporären Tabellenbereich übereinstimmt.

Das Laden beginnt an einer Blockgrenze, so dass sich Ladeoperationen am besten für Daten, die zu neuen Zellen gehören, oder zum ersten Füllen einer Tabelle mit Daten eignen.

Überlegungen zur Protokollierung bei MDC-Tabellen:

In Fällen, in denen Spalten, die zuvor oder auf andere Weise durch Satz-ID-Indizes indiziert waren, nun Dimensionen sind und so durch Blockindizes indiziert werden, verringert sich der Aufwand für Pflege und Protokollierung von Indizes erheblich. Nur wenn der letzte Datensatz in einem gesamten Block gelöscht wird, muss DB2 UDB die BID aus den Blockindizes entfernen und diese Indexoperation protokollieren. Analog braucht DB2 UDB nur dann eine BID in die Blockindizes einzufügen und diese Operation zu protokollieren, wenn ein Datensatz in einen neuen Block eingefügt wird (d. h. wenn es sich um den ersten Datensatz einer

logischen Zelle bzw. eine Einfügung in eine logische Zelle handelt, deren Blöcke zurzeit voll sind). Da Blöcke eine Größe zwischen 2 und 256 Datensatzseiten haben können, ist dieser Pflege- und Protokollierungsaufwand für Blockindizes relativ gering. Einfüge- und Löschoperationen an der Tabelle und den Satz-ID-Indizes werden weiterhin protokolliert.

Überlegungen zu Blockindizes für MDC-Tabellen:

Wenn Sie Dimensionen für eine MDC-Tabelle definieren, werden Dimensionsblockindizes erstellt. Darüber hinaus kann auch ein zusammengesetzter Blockindex erstellt werden, wenn mehrere Dimensionen definiert werden. Wenn Sie hingegen nur eine Dimension für Ihre MDC-Tabelle definiert haben, erstellt DB2 UDB nur einen Blockindex, der in den beiden Funktionen als Dimensionsblockindex und als zusammengesetzter Blockindex verwendet wird. Wenn Sie eine MDC-Tabelle mit Dimensionen in Spalte A und (Spalte A, Spalte B) erstellen, erstellt DB2 UDB analog einen Dimensionsblockindex für Spalte A und einen Dimensionsblockindex für 'Spalte A, Spalte B'. Da ein zusammengesetzter Blockindex ein Blockindex aller Dimensionen in der Tabelle ist, dient der Dimensionsblockindex für 'Spalte A, Spalte B' gleichzeitig als zusammengesetzter Blockindex für die Tabelle.

Der zusammengesetzte Blockindex dient außerdem zur Verarbeitung von Abfragen, bei denen auf Daten in der Tabelle zugegriffen wird, die bestimmte Dimensionswerte haben. Zu beachten ist, dass die Reihenfolge der Schlüsselbestandteile im zusammengesetzten Blockindex seine Verwendung oder Anwendbarkeit bei der Abfrageverarbeitung beeinflussen kann. Die Reihenfolge der Schlüsselbestandteile wird durch die Reihenfolge der Spalten festgelegt, die in der gesamten zur Erstellung der MDC-Tabelle verwendeten Klausel ORGANIZE BY DIMENSIONS vorgefunden wird. Eine Tabelle wird zum Beispiel mit der folgenden Anweisung erstellt:

```
CREATE TABLE t1 (c1 int, c2 int, c3 int, c4 int)
   ORGANIZE BY DIMENSIONS (c1, c4, (c3,c1), c2)
```

In diesem Fall wird der zusammengesetzte Blockindex für die Spalten (c1,c4,c3,c2) erstellt. Beachten Sie, dass die Spalte c1 zwar zweimal in der Dimensionsklausel auftritt, jedoch nur einmal als Schlüsselbestandteil für den zusammengesetzten Blockindex und hier in der Reihenfolge ihres ersten Auftretens verwendet wird. Die Reihenfolge der Schlüsselbestandteile im zusammengesetzten Blockindex macht für die Verarbeitung von Einfügeoperationen keinen Unterschied, kann sich hingegen auf die Verarbeitung von Abfragen auswirken. Wenn also die Spaltenfolge (c1,c2,c3,c4) im zusammengesetzten Blockindex vorgezogen wird, sollte die Tabelle mit folgender Anweisung erstellt werden:

```
CREATE TABLE t1 (c1 int, c2 int, c3 int, c4 int)
   ORGANIZE BY DIMENSIONS (c1, c2, (c3,c1), c4)
```

Zugehörige Konzepte:

- „Indexes" in *SQL Reference, Volume 1*
- „Hinweise zum mehrdimensionalen Clustering" in *Dienstprogramme für das Versetzen von Daten Handbuch und Referenz*
- „Entwerfen von Tabellen mit mehrdimensionalem Clustering (MDC)" auf Seite 177
- „Tabellen- und Indexverwaltung für MDC-Tabellen" in *Systemverwaltung: Optimierung*
- „Optimierungsstrategien für MDC-Tabellen" in *Systemverwaltung: Optimierung*
- „Erstellung, Platzierung und Verwendung von Tabellen mit mehrdimensionalem Clustering (MDC)" auf Seite 186

Zugehörige Referenzen:

- „Sperrmodi für Tabellen- und Satz-ID-Indexsuchen für MDC-Tabellen" in *Systemverwaltung: Optimierung*
- „Sperren für Blockindexsuchen für MDC-Tabellen" in *Systemverwaltung: Optimierung*

Entwerfen von Tabellen mit mehrdimensionalem Clustering (MDC)

Wenn Sie sich einmal zur Arbeit mit MDC-Tabellen entschlossen haben, hängen die auszuwählenden Dimensionen nicht nur vom Typ der Abfragen ab, die auf die Tabellen ausgeführt werden und von einem Clustering auf Blockebene profitieren, sondern, was wesentlich wichtiger ist, von der Menge und der Verteilung Ihrer tatsächlichen Daten. Im Folgenden werden diese Aspekte des Entwurfs von MDC-Tabellen näher erläutert, um Ihnen eine Hilfestellung zur Auswahl der geeigneten Dimensionen und Blockgrößen zu geben.

Abfragen, die von MDC profitieren:

Die erste Überlegung bei der Auswahl von Clusteringdimensionen für Ihre Tabelle besteht in der Bestimmung der Abfragen, die vom Clustering auf Blockebene profitieren sollen. In der Regel werden bei der Auswahl von Dimensionen auf der Grundlage der Abfragen, die die zu erledigende Arbeit an den Daten darstellen, mehrere Kandidaten in Betracht kommen. Die Rangfolge dieser Kandidaten spielt eine wichtige Rolle. Spalten, insbesondere solche mit niedrigen Kardinalitäten, die in Abfragen mit Gleichheits- oder Bereichsvergleichselementen verwendet werden, ziehen den größten Nutzen aus Clusteringdimensionen und sollten daher als Kandidaten für Dimensionen betrachtet werden. Außerdem sollten Sie eine Erstellung von Dimensionen für Fremdschlüssel in einer MDC-Fakttabelle in Betracht ziehen, die in Sternverknüpfungen mit Dimensionstabellen einbezogen werden. Sie sollten die Leistungsvorteile des automatischen und kontinuierlichen Clustering in mehreren Dimensionen sowie des Clustering in EXTENTSIZE großen Speicherbereichen oder Blöcken nicht außer Acht lassen.

Viele Abfragen können von einem mehrdimensionalen Clustering profitieren. Im Folgenden werden Beispiele solcher Abfragen erläutert. Stellen Sie sich für einige der Beispiele vor, dass eine MDC-Tabelle t1 mit den Dimensionen c1, c2 und c3 vorhanden ist. In anderen Beispielen wird eine MDC-Tabelle 'mdctable' mit den Dimensionen 'Farbe' und 'Land' angenommen.

Beispiel 1:

```
SELECT .... FROM t1 WHERE c3 < 5000
```

Diese Abfrage enthält ein Bereichsvergleichselement für eine einzelne Dimension, so dass sie intern umgeschrieben werden kann, um den Zugriff auf die Tabelle über den Dimensionsblockindex für Spalte c3 durchzuführen. Der Index wird nach Block-IDs (BIDs) von Schlüsseln durchsucht, die Werte kleiner 5000 haben. Auf die resultierende Gruppe von Blöcken wird eine relationale Minisuche angewendet, um die tatsächlichen Datensätze abzurufen.

Beispiel 2:

```
SELECT .... FROM t1 WHERE c2 IN (1,2037)
```

Diese Abfrage enthält ein Vergleichselement IN für eine einzelne Dimension und kann Suchen auf der Grundlage eines Blockindex auslösen. Diese Abfrage kann

intern so umgeschrieben werden, dass auf die Tabelle über den Dimensionsblockindex für Spalte c2 zugegriffen wird. Der Index wird nach BIDs von Schlüsseln durchsucht, die die Werte 1 und 2037 besitzen. Auf die resultierende Gruppe von Blöcken wird eine relationale Minisuche durchgeführt, um die tatsächlichen Datensätze abzurufen.

Beispiel 3:

```
SELECT * FROM MDCTABLE WHERE FARBE='BLAU' AND LAND='USA'
```

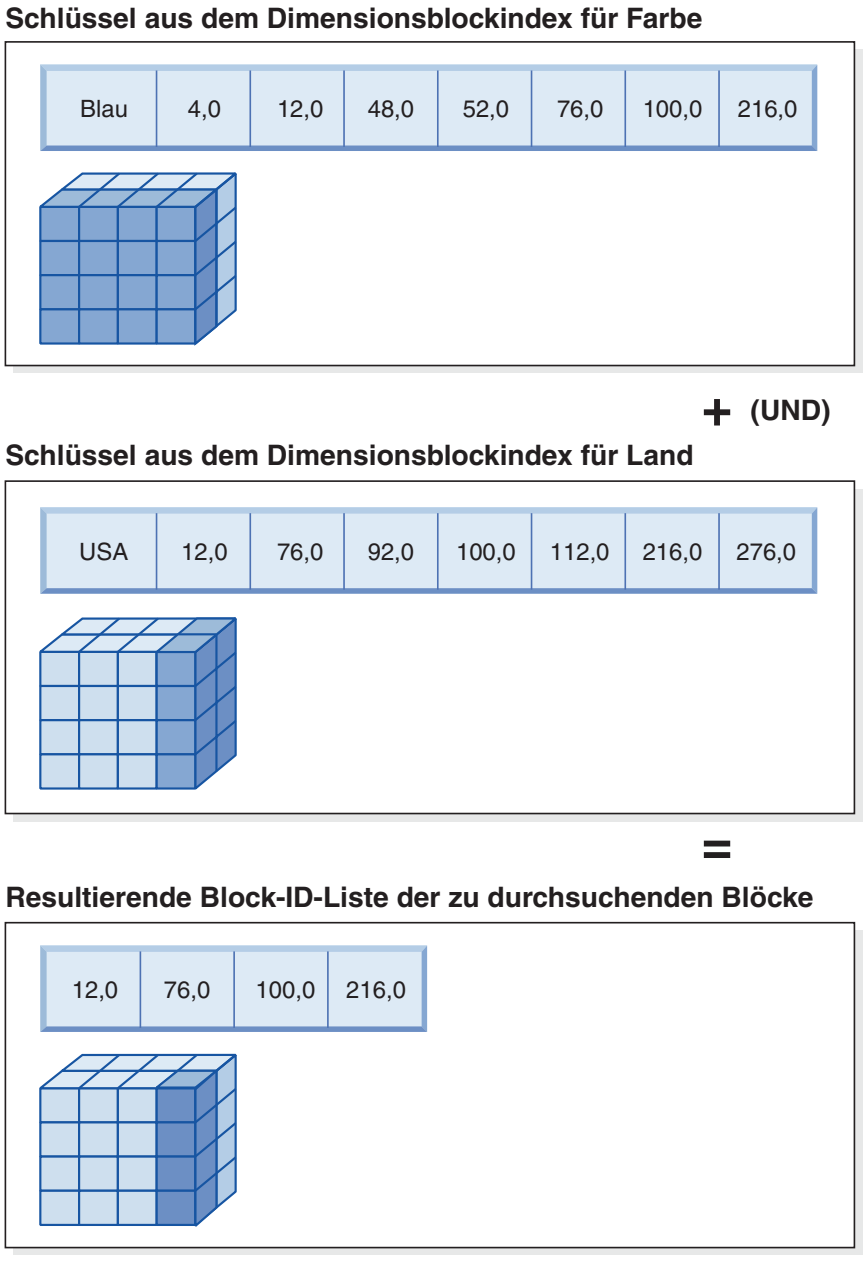


Abbildung 55. Eine Abfrageanforderung mit einer logischen UND-Operation und zwei Blockindizes

Diese Abfrageanforderung wird wie folgt ausgeführt (siehe Abb. 55 auf Seite 178):

- Es werden Suchen in den Dimensionsblockindizes ausgeführt: eine für den Sektor 'Blau' und eine weitere für den Sektor 'USA'.
- Eine logische UND-Operation wird auf die Blöcke angewendet, um die Schnittmenge der beiden Sektoren zu bestimmen. Das heißt, die logische UND-Operation bestimmt nur diejenigen Blöcke, die sich in beiden Sektoren befinden.
- In den resultierenden Blöcken der Tabelle wird eine relationale Minisuche ausgeführt.

Beispiel 4:

```
SELECT ... FROM t1
   WHERE c2 > 100 AND c1 = '16/03/1999' AND c3 > 1000 AND c3 < 5000
```

Diese Abfrage enthält Bereichsvergleichselemente für c2 und c3, ein Gleichheitsvergleichselement für c1 sowie eine logische UND-Operation. Diese Abfrage kann intern so umgeschrieben werden, dass auf die Tabelle über jeden der Dimensionsblockindizes zugegriffen wird:

- Eine Suche im Blockindex für c2 wird ausgeführt, um BIDs von Schlüsseln mit Werten größer als 100 zu finden.
- Eine Suche im Blockindex für c3 wird ausgeführt, um BIDs von Schlüsseln mit Werten zwischen 1000 und 5000 zu finden.
- Eine Suche im Blockindex für c1 wird ausgeführt, um BIDs von Schlüsseln mit dem Wert '16/03/1999' zu finden.

Anschließend wird eine logische UND-Operation auf die resultierenden BIDs aus den einzelnen Blocksuchen angewendet, um deren Schnittmenge zu ermitteln. Und schließlich wird eine relationale Minisuche auf die resultierende Gruppe von Blöcken angewendet, um die tatsächlichen Datensätze zu finden.

Beispiel 5:

```
SELECT * FROM MDCTABLE WHERE FARBE='BLAU' OR LAND='USA'
```

Diese Abfrageanforderung wird wie folgt ausgeführt (siehe Abb. 50 auf Seite 169):

- Es werden Suchen in den Dimensionsblockindizes ausgeführt: eine für jeden Sektor.
- Eine logische ODER-Operation wird angewendet, um die Vereinigungsmenge der beiden Sektoren zu bestimmen.
- In den resultierenden Blöcken der Tabelle wird eine relationale Minisuche ausgeführt.

Beispiel 6:

```
SELECT .... FROM t1 WHERE c1 < 5000 OR c2 IN (1,2,3)
```

Diese Abfrage enthält ein Vergleichselement für die Dimension c1 und ein Vergleichselement IN für die Dimension c2 sowie eine logische ODER-Operation. Diese Abfrage kann intern so umgeschrieben werden, dass auf die Tabelle über die Dimensionsblockindizes für c1 und c2 zugegriffen wird. Im Dimensionsblockindex für c1 wird eine Suche nach den Werten kleiner 5000 und im Dimensionsblockindex für c2 eine Suche nach den Werten 1, 2 und 3 durchgeführt. Auf die resultierenden BIDs aus den Blockindexsuchen wird eine logische ODER-Operation angewendet. Anschließend wird eine relationale Minisuche auf die resultierende Gruppe von Blöcken angewendet, um die tatsächlichen Datensätze zu ermitteln.

Beispiel 7:

```
SELECT .... FROM t1 WHERE c1 = 15 AND c4 < 12
```

Diese Abfrage enthält ein Gleichheitsvergleichselement für die Dimension c1 und ein weiteres Bereichsvergleichselement für eine Spalte, die keine Dimension ist, sowie eine logische UND-Operation. Diese Abfrage kann intern so umgeschrieben werden, dass auf den Dimensionsblockindex für c1 zugegriffen wird, um die Liste der Blöcke aus dem Sektor der Tabelle abzurufen, der für c1 den Wert 15 aufweist. Wenn ein Satz-ID-Index für Spalte c4 vorhanden ist, kann eine Indexsuche durchgeführt werden, um die Satz-IDs von Datensätzen abzurufen, in denen der Wert von c4 kleiner 12 ist. Anschließend kann die resultierende Gruppe von Blöcken durch eine logische UND-Operation mit dieser Datensatzliste verknüpft werden. Durch die Ermittlung dieser Schnittmenge werden die Satz-IDs eliminiert, die nicht in den Blöcken mit c1 gleich 15 vorhanden sind, und nur die aufgelisteten Satz-IDs, die in den entsprechenden Blöcken gefunden werden, werden aus der Tabelle abgerufen.

Wenn kein Satz-ID-Index für c4 vorhanden ist, kann der Blockindex nach den entsprechenden Blöcken durchsucht und während der relationalen Minisuche in den einzelnen Blöcken das Vergleichselement c4 < 12 auf jeden gefundenen Datensatz angewendet werden.

| Beispiel 8:

| In einem Szenario mit den Dimensionen Farbe, Jahr und Land sowie mit einem
| Satz-ID-Index (RID-Index) für die Teilenummer wäre die folgende Abfrage mög-
| lich:

```
| SELECT * FROM MDCTABLE WHERE FARBE='BLAU' AND TEILENUMMER < 1000
```

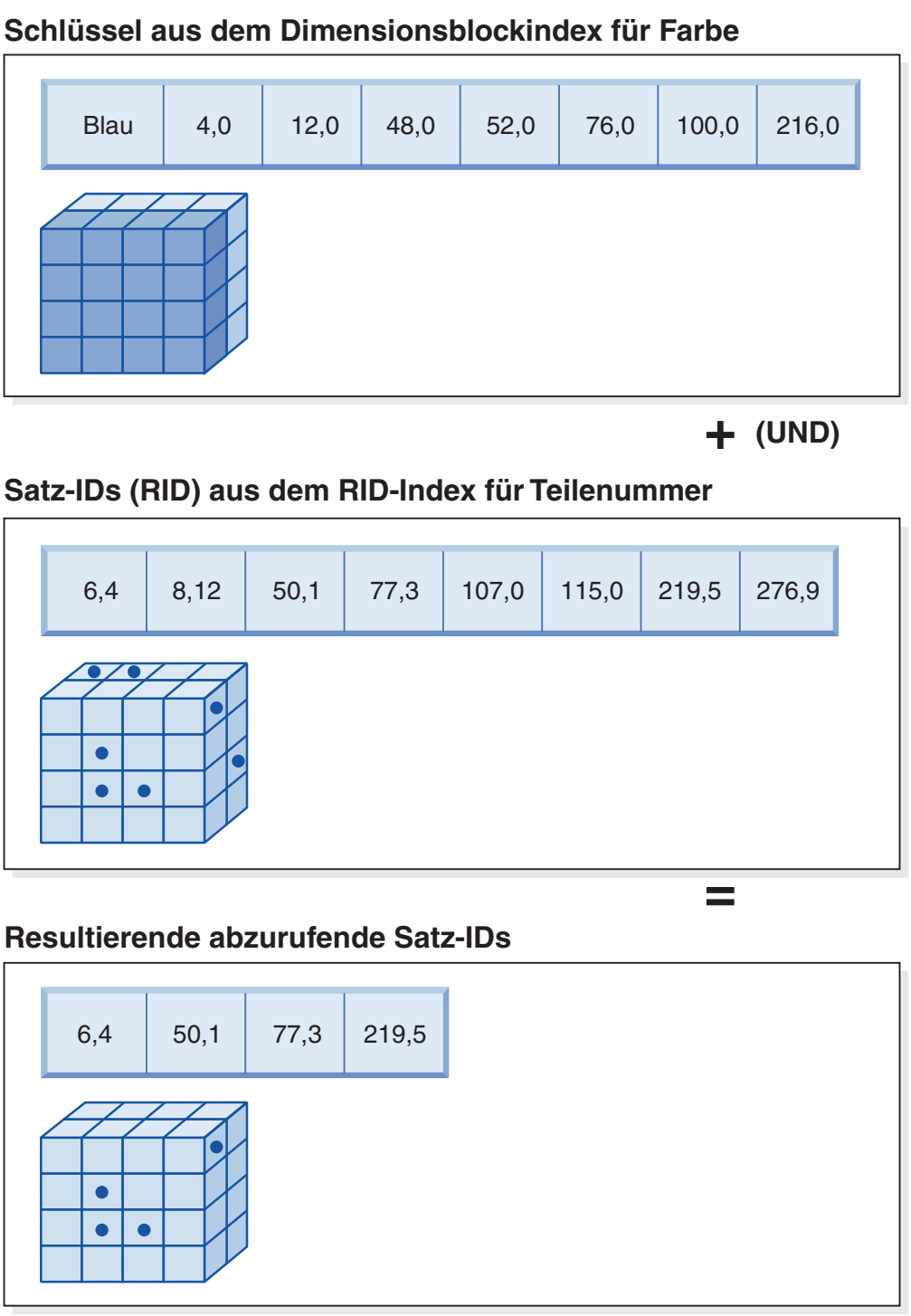


Abbildung 56. Eine Abfrageanforderung mit einer logischen UND-Operation sowie einem Blockindex und einem Satz-ID-Index

Diese Abfrageanforderung wird wie folgt ausgeführt (siehe Abb. 56):

- Es wird eine Suche im Dimensionsblockindex und eine Suche im RID-Index ausgeführt.
- Eine logische UND-Operation wird auf die Blöcke und Satz-IDs angewendet, um die Schnittmenge des Sektors und der Datensätze zu ermitteln, die der Bedingung des Vergleichselements entsprechen.

- Das Ergebnis sind nur die Satz-IDs (RIDs), die zu den qualifizierten Blöcken gehören.

Beispiel 9:

```
SELECT * FROM MDCTABLE WHERE FARBE='BLAU' OR TEILENUMMER < 1000
```

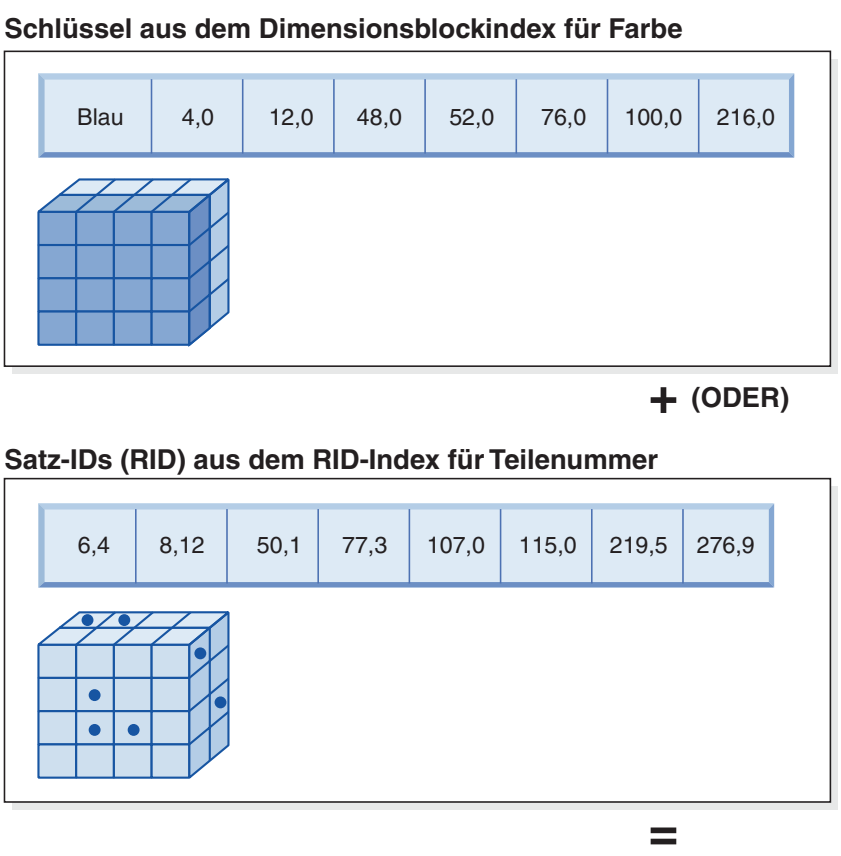


Abbildung 57. Funktionsweise eines Blockindex und eines Satz-ID-Index mit einer logischen ODER-Operation

Diese Abfrageanforderung wird wie folgt ausgeführt (siehe Abb. 57 auf Seite 182):
- Es wird eine Suche im Dimensionsblockindex und eine Suche im RID-Index ausgeführt.
- Eine logische UND-Operation wird auf die Blöcke und Satz-IDs angewendet, um die Vereinigungsmenge des Sektors und der Datensätze zu ermitteln, die der Bedingung des Vergleichselements entsprechen.
- Das Ergebnis umfasst alle Datensätze in den qualifizierten Blöcken und die zusätzlichen Satz-IDs, die außerhalb der qualifizierten Blöcke liegen, jedoch die Bedingung des Vergleichselements erfüllen. In den Blöcken wird eine relationale Minisuche ausgeführt, um deren Datensätze abzurufen, während die zusätzlichen Datensätze außerhalb dieser Blöcke einzeln abgerufen werden.

Beispiel 10:

```
SELECT ... FROM t1 WHERE c1 < 5 OR c4 = 100
```

Diese Abfrage enthält ein Bereichsvergleichselement für Dimension c1 und ein Gleichheitsvergleichselement für Spalte c4, die keine Dimension ist, sowie eine logische ODER-Operation. Wenn ein Satz-ID-Index für Spalte c4 vorhanden ist, kann diese Abfrage intern so umgeschrieben werden, dass eine logische ODER-Operation mit dem Dimensionsblockindex für c1 und dem Satz-ID-Index für c4 durchgeführt wird. Wenn kein Index für c4 vorhanden ist, kann stattdessen eine Tabellensuche gewählt werden, da alle Datensätze überprüft werden müssen. Bei der logischen ODER-Operation würde eine Blockindexsuche auf c1 nach Werten kleiner 4 sowie eine Satz-ID-Indexsuche auf c4 nach Werten gleich 100 durchgeführt. Anschließend wird eine relationale Minisuche in allen qualifizierten Blöcken durchgeführt, da alle Datensätze in diesen Blöcken der Suchbedingung entsprechen, und alle weiteren Satz-IDs für Datensätze außerhalb dieser Blöcke werden ebenfalls abgerufen.

Beispiel 11:

```
SELECT .... FROM t1,d1,d2,d3
   WHERE t1.c1 = d1.c1 and d1.region = 'NY'
      AND t2.c2 = d2.c3 and d2.jahr='1994'
      AND t3.c3 = d3.c3 and d3.produkt='basketball'
```

Diese Abfrage enthält eine Sternverknüpfung. In diesem Beispiel ist t1 die Fakttabelle und hat die Fremdschlüssel c1, c2 und c3, die den Primärschlüsseln der Dimensionstabellen d1, d2 und d3 entsprechen. Die Dimensionstabellen brauchen keine MDC-Tabellen zu sein. Region, Jahr und Produkt sind Spalten der jeweiligen Dimensionstabellen, die mit Hilfe von normalen Indizes oder Blockindizes (wenn die Dimensionstabellen MDC-Tabellen sind) indiziert werden können. Wenn auf eine Fakttabelle über die Werte der Spalten c1, c2 und c3 zugegriffen wird, können Blockindexsuchen in den Dimensionsblockindizes für diese Spalten durchgeführt und anschließend eine logische UND-Operation auf die resultierenden BIDs angewendet werden. Wenn eine Liste von Blöcken vorhanden ist, kann eine relationale Minisuche in jedem Block durchgeführt werden, um die Datensätze abzurufen.

Zellendichte:

Die Entscheidungen, die für die geeigneten Dimensionen sowie für den Wert von EXTENTSIZE getroffen werden, sind von **kritischer** Bedeutung für den MDC-Entwurf. Diese Faktoren bestimmen die erwartete Zelldichte der Tabelle. Sie spielen eine wichtige Rolle, da für jede vorhandene Zelle unabhängig von der Anzahl der Datensätze in der Zelle ein EXTENTSIZE großer Speicherbereich zugeordnet wird. Die richtigen Entscheidungen ermöglichen eine vorteilhafte Nutzung der blockbasierten Indexierung und des mehrdimensionalen Clustering, so dass sich

Leistungsgewinne realisieren lassen. Das Ziel besteht darin, dicht gefüllte Blöcke zu haben, um den größtmöglichen Vorteil aus dem mehrdimensionalen Clustering zu ziehen und eine optimale Speicherplatznutzung zu erhalten.

Aus diesem Grund stellt die Dichte der Zellen in der Tabelle, die aufgrund der aktuellen und für die Zukunft absehbaren Daten zu erwarten ist, einen wichtigen Aspekt beim Entwurf einer mehrdimensionalen Tabelle dar. Sie können eine Gruppe von Dimensionen auf der Grundlage der Abfrageleistung auswählen, die bewirken, dass die Anzahl von Zellen in der Tabelle je nach der Anzahl möglicher Werte für die einzelnen Dimensionen potenziell sehr groß wird. Die Anzahl der möglichen Zellen in der Tabelle entspricht dem kartesischen Produkt der Kardinalitäten jeder der gewählten Dimensionen. Wenn Sie zum Beispiel die Daten der Tabelle nach den Dimensionen Tag, Region und Produkt in Clustern zusammenfassen und die Daten fünf Jahre umfassen, kann Ihre Tabelle `1821 Tage * 12 Regionen * 5 Produkte = 109.260` verschiedene mögliche Zellen enthalten. Jede Zelle, die nur einige wenige Datensätze enthält, benötigt trotzdem einen ganzen Block von Seiten, die ihr zugeordnet sind, um ihre Datensätze zu speichern. Wenn die Blockgröße groß ist, kann die Tabelle schließlich wesentlich größer werden, als sie eigentlich sein müsste.

Verschiedene Entwurfsfaktoren haben Einfluss auf die optimale Zelldichte:

- Variieren der Anzahl von Dimensionen
- Variieren der Granularität einer oder mehrerer Dimensionen
- Variieren der Blockgröße (EXTENTSIZE) und Seitengröße des Tabellenbereichs

Führen Sie die folgenden Schritte aus, um das optimale Entwurfsergebnis zu erzielen:

1. Ermitteln Sie in Frage kommende Kandidatendimensionen.

   Stellen Sie fest, welche Abfragen von einem Clustering auf Blockebene profitieren würden. Untersuchen Sie die potenzielle Auslastung für Spalten, die einige oder alle der folgenden Merkmale besitzen:

   - Bereich oder Gleichheit in Vergleichselementen mit IN-Listen
   - Ein- oder Auslagerung von Daten
   - GROUP BY- oder ORDER BY-Klauseln
   - Verknüpfungsklauseln (insbesondere in Sternschemaumgebungen)

2. Schätzen Sie die Anzahl von Zellen.

   Ermitteln Sie, wie viele potenzielle Zellen in einer Tabelle vorhanden sein können, die nach einer Gruppe von Kandidatendimensionen organisiert ist. Bestimmen Sie die Anzahl der eindeutigen Kombinationen der Dimensionswerte, die in den Daten auftreten. Wenn die Tabelle bereits vorhanden ist, kann die exakte Anzahl für die aktuellen Daten ermittelt werden, indem die Anzahl der unterschiedlichen ('distinct') Werte in jeder der Spalten, die eine Dimension der Tabelle bilden sollen, ausgewählt wird. Alternativ können Sie einen Näherungswert bestimmen, wenn nur die Statistikdaten für eine Tabelle vorliegen, indem Sie die Kardinalitäten der Kandidatenspalten für die geplanten Dimensionen miteinander multiplizieren.

   Anmerkung: Wenn Ihre Tabelle in einer Umgebung mit partitionierten Datenbanken angelegt ist und der Partitionierungsschlüssel keinen Bezug zu einer der in Betracht gezogenen Dimensionen hat, müssen Sie eine Durchschnittsmenge von Daten pro Zelle ermitteln, indem Sie die gesamten Daten durch die Anzahl der Partitionen dividieren.

3. Schätzen Sie die Speicherbelegung oder Dichte ab.

   Im Durchschnitt gehen Sie von einer Zelle aus, die einen zum Teil gefüllten Block besitzt, in dem nur wenige Datensätze gespeichert sind. Die Anzahl teilweise gefüllter Blöcke wird mit abnehmender Anzahl von Datensätzen pro Zelle größer. Beachten Sie außerdem, dass im Durchschnitt (unter der Annahme nur geringer oder keiner Datenabweichung) die Anzahl von Datensätzen pro Zelle durch Dividieren der Anzahl der Datensätze in der Tabelle durch die Anzahl von Zellen ermittelt werden kann. Wenn sich Ihre Tabelle jedoch in einer Umgebung mit partitionierten Datenbanken befindet, müssen Sie berücksichtigen, wie viele Datensätze pro Zelle in jeder Partition vorhanden sind, da Blöcke für Daten auf Partitionsebene zugeordnet werden. Bei der Schätzung der Speicherplatzbelegung und der Dichte in einer Umgebung mit Datenpartitionen müssen Sie die durchschnittliche Anzahl von Datensätzen pro Zelle in jeder Partition, und nicht für die gesamte Tabelle, in Betracht ziehen. Weitere Informationen finden Sie im Abschnitt „Erstellung, Platzierung und Verwendung von Tabellen mit mehrdimensionalem Clustering (MDC)".

   Es gibt verschiedene Möglichkeiten zur Verbesserung der Dichte:

   - Verringern Sie die Blockgröße, so dass teilweise gefüllte Blöcke weniger Speicherplatz belegen.

     Verringern Sie die Größe der einzelnen Blöcke, indem Sie einen möglichst kleinen geeigneten Wert für EXTENTSIZE wählen. Jede Zelle, die einen teilweise gefüllten Block enthält oder die nur einen Block mit wenigen Datensätzen enthält, verschwendet Speicherplatz. Andererseits benötigen nun Zellen, die viele Datensätze enthalten, mehr Blöcke, um diese aufzunehmen. Infolgedessen erhöht sich die Anzahl von Block-IDs (BIDs) für diese Zellen in den Blockindizes, so dass diese Indizes größer werden und potenziell mehr Einfüge- und Löschoperationen an diesen Indizes verursachen, da Blöcke schneller geleert oder gefüllt werden. Darüber hinaus führt dies zu einer größeren Zahl kleiner Datencluster in der Tabelle für diese volleren Zellwerte im Gegensatz zu einer kleineren Anzahl größerer Datencluster.
   - Verringern Sie die Anzahl von Zellen, indem Sie die Anzahl der Dimensionen verkleinern oder die Granularität der Zellen mit einer generierten Spalte erhöhen.

     Sie können eine oder mehrere Dimensionen auf eine geringere Granularität hochstufen, um ihnen eine geringere Kardinalität zu verleihen. Zum Beispiel können Sie die Daten im vorigen Beispiel weiterhin in den Dimensionen 'Region' und 'Produkt' in Clustern zusammenfassen, die Dimension 'Tag' jedoch durch eine Dimension 'JahrUndMonat' ersetzen. Dies ergäbe Kardinalitäten von 60 (12 Monate mal 5 Jahre) 12 und 5 für 'JahrUndMonat', 'Region' und 'Produkt' mit einer Anzahl möglicher Zellen von 3600. Jede Zelle umfasst dann einen größeren Bereich von Werten und enthält mit geringerer Wahrscheinlichkeit nur wenige Datensätze.

     Darüber hinaus sollten Sie auch die Vergleichselemente in Ihre Überlegungen mit einbeziehen, die am häufigsten auf die beteiligten Spalten angewendet werden, und berücksichtigen, ob sich viele von ihnen zum Beispiel auf Spalten mit Monatsdatums-, Quartals- oder Tagesangaben beziehen. Dies hat Einfluss auf die Entscheidung, ob die Granularität der Dimension geändert werden sollte. Wenn zum Beispiel die meisten Vergleichselemente bestimmte Tage betreffen und Sie die Tabelle nach der Spalte 'Monat' zu Clustern zusammengefasst haben, kann DB2® Universal Database (DB2 UDB) mit Hilfe des Blockindex für 'JahrUndMonat' rasch die Monate eingrenzen, in denen die gewünschten Tage enthalten sind, und nur auf die ihnen zugeordneten Blöcke zugreifen. Beim Durchsuchen der Blöcke muss das Tagesvergleichselement jedoch angewendet werden, um die gesuchten Tage zu

bestimmen. Wenn Sie allerdings nach 'Tag' in Clustern zusammenfassen (und dies bedeutet eine hohe Kardinalität), kann der Blockindex für 'Tag' dazu verwendet werden, die zu durchsuchenden Blöcke zu ermitteln, und das Vergleichselement für den Tag muss nur jeweils auf den ersten Datensatz in jeder gefundenen Zelle angewendet werden. In diesem Fall kann es vorteilhafter sein, eine der anderen Dimensionen mit einer geringeren Granularität zu definieren, um die Dichte der Zellen zu erhöhen. Zum Beispiel könnte die Spalte 'Region', die zwölf verschiedene Werte enthält, mit Hilfe einer benutzerdefinierten Funktion auf die Regionen 'West', 'Nord', 'Süd' und 'Ost' hochgestuft werden.

Zugehörige Konzepte:

- „Der Designadvisor" in *Systemverwaltung: Optimierung*
- „Tabellen mit mehrdimensionalem Clustering" auf Seite 158
- „Erstellung, Platzierung und Verwendung von Tabellen mit mehrdimensionalem Clustering (MDC)" auf Seite 186

Erstellung, Platzierung und Verwendung von Tabellen mit mehrdimensionalem Clustering (MDC)

Bei der Erstellung von MDC-Tabellen sind zahlreiche Faktoren zu beachten. In den folgenden Abschnitten wird erläutert, wie Ihre Entscheidungen zur Erstellung, Platzierung und Verwendung Ihrer MDC-Tabellen durch Ihre aktuelle Datenbankumgebung (z. B. ob es sich um eine partitionierte Datenbank handelt oder nicht) sowie durch Ihre Auswahl der Dimensionen für die einzelnen MDC-Tabellen beeinflusst werden können. Darüber hinaus wird der DB2®-Designadvisor und seine Verwendung zur Generierung von Empfehlungen zu diesen Aspekten behandelt.

Versetzen von Daten aus einer vorhandenen Tabelle in eine MDC-Tabelle:

Zur Verbesserung der Abfrageleistung und zur Verringerung des Aufwands für Datenpflegeoperationen in einem Data Warehouse oder einer großen Datenbankumgebung können Sie Daten aus regulären Tabellen in Tabellen mit mehrdimensionalem Clustering (MDC) versetzen. Zum Versetzen von Daten aus einer vorhandenen Tabelle in eine MDC-Tabelle exportieren Sie Ihre Daten, löschen die ursprüngliche Tabelle (optional), erstellen eine MDC-Tabelle (mit Hilfe der Anweisung CREATE TABLE mit der Klausel ORGANIZE BY DIMENSIONS) und laden Ihre Daten in die MDC-Tabelle.

Eine ALTER TABLE-Prozedur mit dem Namen SYSPROC.ALTOBJ kann zur Ausführung der Umsetzung von Daten aus einer vorhandenen Tabelle in eine MDC-Tabelle verwendet werden. Die Prozedur wird über den DB2-Designadvisor aufgerufen. Die Zeit, die zur Umsetzung der Daten zwischen den Tabellen erforderlich ist, kann enorm sein und hängt von der Größe der Tabelle und der Menge von Daten ab, die umgesetzt werden müssen.

Die Prozedur ALTOBJ führt bei der Änderung einer Tabelle folgende Aktionen aus:

- Sie löscht alle abhängigen Objekte der Tabelle.
- Sie benennt die Tabelle um.
- Sie erstellt die Tabelle mit der neuen Definition.
- Sie erstellt alle abhängigen Objekte der Tabelle erneut.
- Sie setzt vorhandene Daten in der Tabelle in die Daten um, die für die neue Tabelle erforderlich sind. Das heißt, sie wählt die Daten aus der alten Tabelle aus

und lädt sie in die neue Tabelle, wobei Spaltenfunktionen zur Konvertierung eines alten Datentyps in einen neuen Datentyp eingesetzt werden können.

MDC-Tabellen in SMS-Tabellenbereichen:

Wenn Sie planen, MDC-Tabellen in einem SMS-Tabellenbereich zu speichern, wird die Verwendung der mehrseitigen Dateizuordnung ausdrücklich empfohlen.

Anmerkung: Die mehrseitige Dateizuordnung ist ab Version 8.2 die Standardeinstellung für neu erstellte Datenbanken.

Diese Empfehlung ist darin begründet, dass MDC-Tabellen jeweils um ganze EXTENTSIZE große Speicherbereiche erweitert werden, wobei es wichtig ist, dass alle Seiten in diesen Speicherbereichen physisch aufeinander folgen. Daher lässt sich aus einer Inaktivierung der mehrseitigen Dateizuordnung kein Vorteil erzielen. Vielmehr erhöht eine Aktivierung dieser Funktion die Wahrscheinlichkeit beträchtlich, dass die Seiten in den einzelnen Speicherbereichen physisch zusammenhängend sind.

MDC-Advisorfunktion im DB2-Designadvisor:

Der DB2-Designadvisor (db2advis), bisher als Indexadvisorfunktion bezeichnet, verfügt über eine MDC-Funktion. Diese Funktion empfiehlt Clusteringdimensionen zur Verwendung in einer MDC-Tabelle, einschließlich Vergröberungen (d. h. Verringerungen der Granularität) von Basisspalten, um die Auslastungsleistung zu verbessern. Mit der Bezeichnung *Vergröberung* ist ein mathematischer Ausdruck zur Verringerung der Anzahl unterschiedlicher Werte in einer Clusteringdimension gemeint. Ein allgemeines Beispiel für eine Vergröberung ist die Verringerung der Kardinalität für das Datum, bei dem die Anzahl unterschiedlicher Werte mit der Verringerung der Granularität von Datum auf Woche, Monat oder Quartal des Jahres abnimmt.

Die Empfehlung schließt die Ermittlung potenzieller generierter Spalten mit ein, die eine Vergröberung von Dimensionen definieren. Die Empfehlung bezieht sich nicht auf mögliche Blockgrößen. Bei der Generierung von Empfehlungen für MDC-Tabellen wird der Wert für EXTENTSIZE des Tabellenbereichs zugrunde gelegt. Die Empfehlung geht von der Annahme aus, dass die empfohlene MDC-Tabelle im gleichen Tabellenbereich wie die vorhandene Tabelle erstellt werden soll, so dass sie den gleichen Wert für EXTENTSIZE haben wird. Die Empfehlungen für MDC-Dimensionen würden sich abhängig von dem Wert für EXTENTSIZE des Tabellenbereichs ändern, da die Speicherbereichsgröße Auswirkungen auf die Anzahl von Datensätzen hat, die in einen Block oder eine Zelle passen. Dies hat unmittelbaren Einfluss auf die Dichte der Zellen.

Lediglich Dimensionen, die nur aus einer Spalte, nicht jedoch aus mehreren Spalten zusammengesetzt sind, werden in Betracht gezogen, obwohl einzelne oder mehrere Dimensionen für die Tabelle empfohlen werden können. Die MDC-Funktion empfiehlt Vergröberungen für die meisten unterstützten Datentypen, um die Kardinalität von Zellen in der resultierenden MDC-Lösung zu reduzieren. Ausnahmen bilden die Datentypen CHAR, VARCHAR, GRAPHIC und VARGRAPH. Alle unterstützten Datentypen werden in INTEGER umgesetzt und ihre Kardinalität durch einen generierten Ausdruck verringert.

Das Ziel der MDC-Funktion des DB2-Designadvisors besteht darin, MDC-Lösungen auszuwählen, die zu einer Leistungssteigerung führen. Ein sekundäres Ziel besteht darin, das Anwachsen des Speicherbedarfs der Datenbank auf einer

moderaten Ebene zu halten. Das maximale Speicherwachstum jeder Tabelle wird mit Hilfe einer statistischen Methode ermittelt.

Die Analyseoperation im Designadvisor berücksichtigt nicht nur die Vorteile des Zugriffs über Blockindizes, sondern auch die Auswirkungen des mehrdimensionalen Clustering auf Einfüge-, Aktualisierungs- und Löschoperationen an Dimensionen der Tabelle. Diese Aktionen an der Tabelle können potenziell bewirken, dass Datensätze zwischen Zellen versetzt werden. Darüber hinaus modelliert die Analyseoperation die potenziellen Leistungsauswirkungen einer Tabellenerweiterung, die sich aus der Organisation von Daten nach bestimmten MDC-Dimensionen ergibt.

Die MDC-Funktion wird über die Markierung `-m <empfehlungstyp>` im Dienstprogramm 'db2advis' aktiviert. Der Empfehlungstyp „C" dient zur Angabe von MDC-Tabellen. Folgende Empfehlungstypen sind verfügbar: „I" für Index, „M" für gespeicherte Abfragetabellen (MQT), „C" für MDC und „P" für Datenbankpartitionierung. Die Empfehlungstypen können miteinander kombiniert werden.

Anmerkung: Der DB2-Designadvisor untersucht keine Tabellen, die weniger als zwölf EXTENTSIZE-Speicherbereiche groß sind.

Der Designadvisor analysiert sowohl gespeicherte Abfragetabellen (MQTs) als auch reguläre Basistabellen, um Empfehlungen vorzuschlagen.

Die Ausgabe aus der MDC-Funktion umfasst folgende Informationen:

- Ausdrücke für generierte Spalten für jede Tabelle zur Vergröberung von Dimensionen, die in der MDC-Lösung auftreten.
- Eine Empfehlung für eine ORGANIZE BY-Klausel für jede Tabelle.

Die Empfehlungen werden sowohl auf der Standardausgabeeinheit (stdout) als auch in den ADVISE-Tabellen ausgegeben, die zur EXPLAIN-Einrichtung gehören.

MDC-Tabellen und Datenbankpartitionierung:

Das mehrdimensionale Clustering kann in Verbindung mit einer Datenbankpartitionierung verwendet werden. Tatsächlich kann MDC eine Datenbankpartitionierung ergänzen. Eine Datenbankpartitionierung dient zur Verteilung von Daten aus einer Tabelle auf mehrere physische oder logische Knoten zu folgenden Zwecken:

- Nutzung der Vorteile mehrerer Maschinen, um die Parallelverarbeitung von Anforderungen zu verbessern
- Physische Vergrößerung der Tabelle über die Grenzen einer einzelnen Partition hinaus
- Verbesserung der Skalierbarkeit der Datenbank

Der Grund zur Partitionierung einer Tabelle ist davon unabhängig, ob die Tabelle eine MDC-Tabelle oder eine reguläre Tabelle ist. Zum Beispiel unterscheiden sich die Regeln zur Auswahl von Spalten für den Partitionierungsschlüssel nicht. Der Partitionierungsschlüssel für eine MDC-Tabelle kann jede beliebige Spalte ohne Rücksicht darauf enthalten, ob die Spalten Teil einer Dimension der Tabelle sind oder nicht.

Wenn der Partitionierungsschlüssel mit einer Dimension aus der Tabelle identisch ist, wird jede Datenbankpartition einen anderen Abschnitt der Tabelle enthalten. Wenn die MDC-Tabelle aus dem erläuterten Beispiel nach Farbe über zwei Partitionen partitioniert wäre, würde die Spalte 'Farbe' zur Teilung der Daten verwendet.

Infolgedessen können zum Beispiel die Sektoren 'Rot' und 'Blau' in einer Partition und der Sektor 'Gelb' in einer anderen Partition abgelegt werden. Wenn der Partitionierungsschlüssel nicht mit den Dimensionen aus der Tabelle identisch ist, wird in jeder Datenbankpartition eine Untergruppe der Daten aus allen Sektoren gespeichert. Bei der Auswahl der Dimensionen und der Abschätzung der Zellbelegung (siehe Abschnitt „Zelldichte") ist zu beachten, dass sich die durchschnittliche Menge der Daten pro Zelle durch Dividieren der gesamten Daten durch die Anzahl der Partitionen bestimmen lässt.

MDC-Tabellen mit mehreren Dimensionen:

Wenn Sie wissen, dass bestimmte Vergleichselemente in Abfragen sehr häufig verwendet werden, können Sie die Tabelle mit Hilfe der Klausel ORGANIZE BY DIMENSIONS nach den beteiligten Spalten in Clustern organisieren.

Beispiel 1:

```
CREATE TABLE T1 (c1 DATE, c2 INT, c3 INT, c4 DOUBLE)
   ORGANIZE BY DIMENSIONS (c1, c3, c4)
```

Die Tabelle in Beispiel 1 wird nach den Werten innerhalb von drei nativen Spalten in Clustern organisiert, so dass ein logischer Würfel (mit drei Dimensionen) gebildet wird. Die Tabelle kann nun bei der Abfrageverarbeitung unter Verwendung einer oder mehrerer dieser Dimensionen logisch in Sektoren untergliedert werden, so dass nur die Blöcke in den entsprechenden Sektoren oder Zellen durch die angewendeten relationalen Operatoren verarbeitet werden. Beachten Sie, dass die Größe eines Blocks (die Anzahl von Seiten) mit dem Wert für EXTENTSIZE der Tabelle übereinstimmt.

MDC-Tabellen mit Dimensionen auf der Basis mehrerer Spalten:

Jede Dimension kann aus einer oder mehreren Spalten bestehen. Zum Beispiel können Sie eine Tabelle erstellen, die nach einer Dimension mit zwei Spalten organisiert ist.

Beispiel 2:

```
CREATE TABLE T1 (c1 DATE, c2 INT, c3 INT, c4 DOUBLE)
   ORGANIZE BY DIMENSIONS (c1, (c3, c4))
```

In Beispiel 2 wird die Tabelle in zwei Dimensionen organisiert: c1 und (c3, c4). Bei der Abfrageverarbeitung kann die Tabelle also logisch in einen Sektor nach der Dimension für c1 oder einen Sektor nach der zusammengesetzten Dimension (c3, c4) untergliedert werden. Die Tabelle enthält die gleiche Anzahl von Blöcken wie die Tabelle in Beispiel 1, jedoch einen Dimensionsblockindex weniger. In Beispiel 1 werden drei Dimensionsblockindizes erstellt: jeweils einer für jede der Spalten c1, c3 und c4. In Beispiel 2 werden zwei Dimensionsblockindizes erstellt: einer für Spalte c1 und der andere für die Spalten c3 und c4. Der Hauptunterschied zwischen diesen beiden Ansätzen besteht darin, dass in Beispiel 1 Abfragen, die nur die Spalte c4 betreffen, den Dimensionsblockindex für c4 nutzen können, um rasch und direkt auf Blöcke mit relevanten Daten zuzugreifen. In Beispiel 2 ist die Spalte c4 der zweite Teil eines Schlüssels in einem Dimensionsblockindex, so dass Abfragen, die nur die Spalte c4 betreffen, mehr Verarbeitungsaufwand erfordern. Allerdings hat DB2 Universal Database™ (DB2 UDB) in Beispiel 2 einen Blockindex weniger zu verwalten und zu speichern.

Der DB2-Designadvisor schlägt keine Empfehlungen für Dimensionen aus mehr als einer Spalte vor.

MDC-Tabellen mit Spaltenausdrücken als Dimensionen:

Spaltenausdrücke können ebenfalls zum Clustering von Dimensionen verwendet werden. Die Möglichkeit, Daten nach Spaltenausdrücken in Clustern zu organisieren, ist nützlich, wenn Dimensionen in eine gröbere Granularität hochgestuft werden sollen, zum Beispiel wenn eine Adresse auf eine geografische Position bzw. Region oder ein Datum auf eine Woche, Monat oder Jahr abgebildet wird. Zur Implementierung der Hochstufung von Dimensionen auf diese Weise können Sie generierte Spalten verwenden. Dieser Typ von Spaltendefinition ermöglicht die Erstellung von Spalten durch Ausdrücke, die Dimensionen darstellen können. In Beispiel 3 erstellt die Anweisung eine Tabelle, die nach einer Basisspalte und zwei Spaltenausdrücken in Clustern organisiert wird.

Beispiel 3:

```
CREATE TABLE T1(c1 DATE, c2 INT, c3 INT, c4 DOUBLE,
   c5 DOUBLE GENERATED ALWAYS AS (c3 + c4),
   c6 INT GENERATED ALWAYS AS (MONTH(C1))
      ORGANIZE BY DIMENSIONS (c2, c5, c6)
```

In Beispiel 3 ist Spalte c5 ein Ausdruck, der auf den Spalten c3 und c4 basiert, während die Spalte c6 die Spalte c1 in eine grobere Granularität der Zeit hochstuft. Diese Anweisung organisiert die Tabelle in Clustern nach den Werten in den Spalten c2, c5 und c6.

Bereichsabfragen für eine Dimension auf einer generierten Spalte erfordern monotone Spaltenfunktionen:

Ausdrücke müssen monoton sein, damit Bereichsvergleichselemente für Dimensionen auf generierten Spalten abgeleitet werden können. Wenn Sie eine Dimension auf einer generierten Spalte erstellen, können Abfragen auf die Basisspalte den Blockindex für die generierte Spalte nutzen, um die Leistung zu verbessern, allerdings mit einer Ausnahme. Für Bereichsabfragen auf der Basisspalte (z. B. Datum) muss der Ausdruck, der zur Generierung der Spalte in der Anweisung CREATE TABLE verwendet wird, monoton sein, um eine Bereichssuche im Dimensionsblockindex verwenden zu können. Obwohl ein Spaltenausdruck jeden gültigen Ausdruck (einschließlich benutzerdefinierter Funktionen - UDFs) enthalten kann, können bei nicht monotonen Ausdrücken nur Gleichheitsvergleichselemente und Vergleichselemente mit IN-Listen den Blockindex verwenden, um die Abfrage zu erfüllen, wenn sich diese Vergleichselemente auf die Basisspalte beziehen.

Nehmen Sie zum Beispiel an, dass eine MDC-Tabelle mit Dimensionen auf der generierten Spalte für Monat mit `monat = INTEGER (datum)/100` erstellt wird. Für Abfragen auf der Dimension (monat) können Blockindexsuchen ausgeführt werden. Für Abfragen auf der Basisspalte (datum), können ebenfalls Blockindexsuchen ausgeführt werden, um die zu durchsuchenden Blöcke einzugrenzen. Anschließend können die Vergleichselemente für das Datum nur auf die Datensätze in diesen Blöcken angewendet werden.

Der Compiler generiert zusätzliche Vergleichselemente zur Verwendung bei der Blockindexsuche. Zum Beispiel wird die folgende Abfrage formuliert:

```
SELECT * FROM MDCTABLE WHERE DATUM > "19999/03/03" AND DATUM < "2000/01/15"
```

Der Compiler generiert die folgenden zusätzlichen Vergleichselemente: „monat >= 199903" und „monat < 200001". Diese Vergleichselemente können zur Suche im

Dimensionsblockindex verwendet werden. Beim Durchsuchen der resultierenden Blöcke werden die ursprünglichen Vergleichselemente auf die Datensätze in den Blöcken angewendet.

Ein nicht monotoner Ausdruck würde nur die Anwendung von Gleichheitsvergleichselementen auf diese Dimension zulassen. Ein gutes Beispiel für eine nicht monotone Funktion ist die Funktion MONTH(), wie sie in der Definition von Spalte c6 in Beispiel 3 zu sehen ist. Wenn die Spalte c1 ein Datum, eine Zeitmarke oder eine gültige Zeichenfolgendarstellung eines Datums oder einer Zeitmarke ist, liefert die Funktion einen Ganzzahlwert aus dem Bereich von 1 bis 12. Obwohl die Ausgabe der Funktion deterministisch ist, ähnelt sie einer Schrittfunktion (d. h. einem zyklischen Muster):

```
MONTH(date('99/01/05')) = 1
MONTH(date('99/02/08')) = 2
MONTH(date('99/03/24')) = 3
MONTH(date('99/04/30')) = 4
...
MONTH(date('99/12/09')) = 12
MONTH(date('00/01/18')) = 1
MONTH(date('00/02/24')) = 2
...
```

Obwohl das Datum in diesem Beispiel kontinuierlich fortschreitet, tut dies der Wert der Funktion MONTH(datum) nicht. Dies bedeutet zum Beispiel, dass nicht garantiert ist, dass bei zwei Werten datum1 und datum2, wobei datum1 größer ist als datum2, der Wert von MONTH(datum1) größer als oder gleich dem Wert von MONTH(datum2) ist. Eben dies ist die Monotoniebedingung. Diese Nichteinhaltung der Monotoniebedingung ist zwar zulässig, aber sie beschränkt die Dimension in der Weise, dass ein Bereichsvergleichselement, das die Basisspalte betrifft, kein Bereichsvergleichselement für die Dimension generieren kann. Jedoch ist ein Bereichsvergleichselement für den Ausdruck problemlos möglich, wie zum Beispiel `where month(c1) between 4 and 6`. Die entsprechende Verarbeitung kann den Index für die Dimension mit einem Startschlüsselwert 4 und einem Endschlüsselwert 6 ganz normal verwenden.

Um diese Funktion monoton zu machen, müssten Sie das Jahr als übergeordnete Ordnungskomponente des Monats mit einschließen. DB2 UDB stellt eine Erweiterung zur integrierten Funktion INTEGER() zur Verfügung, um die Definition eines monotonen Ausdrucks für Datum zu unterstützten. INTEGER(datum) liefert eine Ganzzahldarstellung des Datums, die dann dividiert werden kann, um eine Ganzzahldarstellung des Jahres und des Monats zu ermitteln. Zum Beispiel liefert `INTEGER(date('2000/05/24'))` den Wert `20000524`. Somit ist `INTEGER(date('2000/05/24'))/100 = 200005`. Die Funktion INTEGER(datum)/100 ist monoton.

Analog verfügen auch die integrierten Funktionen DECIMAL() und BIGINT() über Erweiterungen, die es Ihnen ermöglichen, monotone Funktionen abzuleiten. Die Funktion DECIMAL(zeitmarke) liefert eine dezimale Darstellung einer Zeitmarke und ihr Wert kann in monotonen Ausdrücken zur Ableitung steigender Werte für Monat, Tag, Stunde, Minute usw. verwendet werden. Die Funktion BIGINT(datum) liefert eine BIGINTEGER-Darstellung des Datums ähnlich wie INTEGER(datum).

DB2 UDB bestimmt die Monotonie eines Ausdrucks, falls möglich, bei der Erstellung der generierten Spalte für die Tabelle oder bei der Erstellung einer Dimension aus einem Ausdruck in der DIMENSIONS-Klausel. Bestimmte Funktionen können als die Monotonie bewahrend erkannt werden, wie zum Beispiel DATENUM( ), DAYS( ), YEAR( ). Außerdem sind verschiedene mathematische Ausdrücke wie

Division, Multiplikation oder Addition einer Spalte und einer Konstante Monotonie bewahrend. In Fällen, in denen DB2 UDB ermittelt, dass ein Ausdruck die Monotonie nicht bewahrt, oder wenn sich dies nicht feststellen lässt, unterstützt die Dimension nur die Verwendung von Gleichheitsvergleichselementen auf der entsprechenden Basisspalte.

Zugehörige Konzepte:

- „Parameter EXTENTSIZE" auf Seite 130
- „Hinweise zum mehrdimensionalen Clustering" in *Dienstprogramme für das Versetzen von Daten Handbuch und Referenz*
- „Tabellen mit mehrdimensionalem Clustering" auf Seite 158
- „Entwerfen von Tabellen mit mehrdimensionalem Clustering (MDC)" auf Seite 177

Zugehörige Tasks:

- „Definieren von Dimensionen für eine Tabelle" in *Systemverwaltung: Implementierung*

Zugehörige Referenzen:

- „CREATE TABLE statement" in *SQL Reference, Volume 2*
- „db2empfa - Enable Multipage File Allocation Command" in *Command Reference*

Kapitel 6. Entwerfen verteilter Datenbanken

Aktualisieren einer Einzeldatenbank in einer Transaktion

Die einfachste Form der Transaktion besteht darin, innerhalb einer Arbeitseinheit lediglich für eine Datenbank Lese- und Schreibvorgänge auszuführen. Diese Art des Datenbankzugriffs wird als *ferne Arbeitseinheit* bezeichnet.

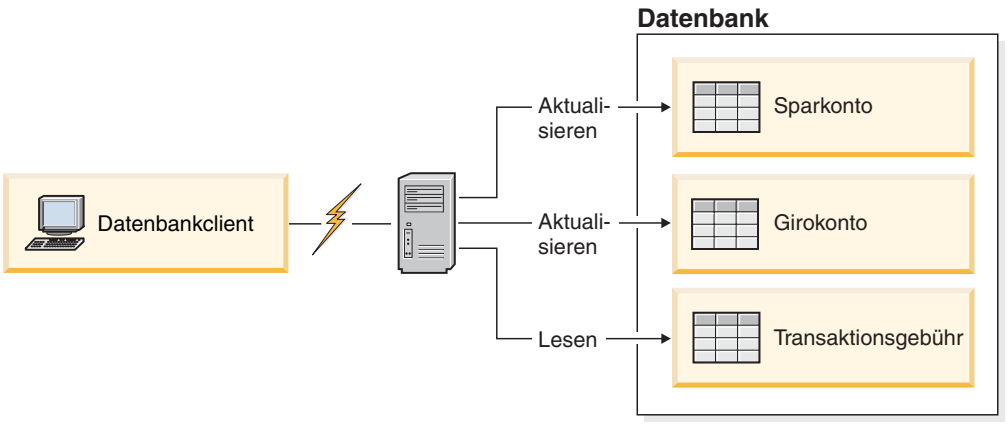


Abbildung 58. Verwenden einer einzelnen Datenbank in einer Transaktion

Abb. 58 zeigt einen Datenbankclient, der eine Geldtransaktionsanwendung ausführt. Diese Anwendung greift auf eine Datenbank zu, die Tabellen für Giro- und Sparkonten sowie einen Plan für die Transaktionsgebühren enthält. Die Anwendung muss folgende Schritte ausführen:

- Akzeptieren des Betrags, der von der Benutzerschnittstelle überwiesen werden soll
- Subtrahieren des Betrags vom Sparkonto und Ermitteln des neuen Kontostands
- Lesen des Gebührenplans, um die Transaktionsgebühr für ein Sparkonto mit dem angegebenen Kontostand zu bestimmen
- Subtrahieren der Transaktionsgebühr vom Sparkonto
- Gutschreiben des Überweisungsbetrags auf dem Girokonto
- Festschreiben der Transaktion (Arbeitseinheit)

Vorgehensweise:

Zur Einrichtung einer solchen Anwendung sind folgende Schritte auszuführen:

1. Erstellen der Tabellen für das Sparkonto, das Girokonto und den Gebührenplan für die Transaktion innerhalb derselben Datenbank
2. Wenn physisch fern, Einrichten des Datenbankservers für die Verwendung des geeigneten Übertragungsprotokolls
3. Wenn physisch fern, Katalogisieren des Knoten und der Datenbank, um die Datenbank auf den Datenbankservern zu identifizieren
4. Vorkompilieren Ihres Anwendungsprogramms, um eine Verbindung des Typs 1 anzugeben, d. h. Angeben von CONNECT 1 (Standardwert) im Befehl PRECOMPILE PROGRAM

Zugehörige Konzepte:

- „Arbeitseinheiten" auf Seite 31

Zugehörige Tasks:

- „Aktualisieren einer Einzeldatenbank in einer Transaktion für mehrere Datenbanken" auf Seite 194
- „Aktualisieren mehrerer Datenbanken in einer Transaktion" auf Seite 195

Zugehörige Referenzen:

- „PRECOMPILE Command" in *Command Reference*

Verwenden mehrerer Datenbank in einer Transaktion

Bei Verwendung mehrerer Datenbanken in einer einzelnen Transaktion gelten andere Anforderungen für das Einrichten und Verwalten Ihrer Umgebung, je nachdem, wie viele Datenbanken in der Transaktion aktualisiert werden.

Aktualisieren einer Einzeldatenbank in einer Transaktion für mehrere Datenbanken

Wenn Ihre Daten über mehrere Datenbanken verteilt sind, möchten Sie eventuell eine Datenbank aktualisieren, während Sie für eine oder mehrere andere Datenbanken Lesevorgänge ausführen. Diese Art des Zugriffs kann innerhalb einer einzigen Arbeitseinheit (Transaktion) erfolgen.

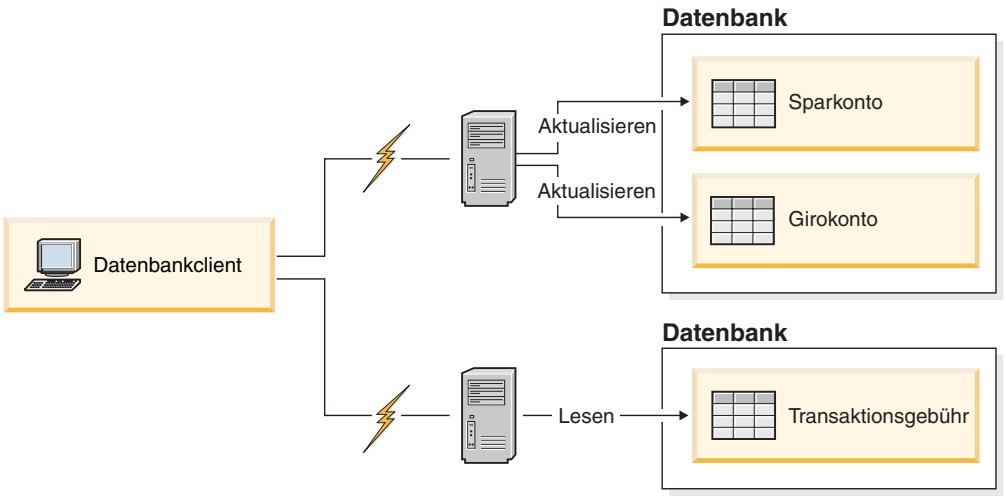


Abbildung 59. Verwenden mehrerer Datenbanken in einer Transaktion

Abb. 59 zeigt einen Datenbankclient, der eine Geldtransaktionsanwendung ausführt, die auf zwei Datenbankserver zugreift: einer enthält das Sparkonto und das Girokonto, der andere den Plan für die Transaktionsgebühren.

Vorgehensweise:

Zur Einrichtung einer Geldtransaktionsanwendung für diese Umgebung sind folgende Schritte auszuführen:

1. Erstellen der erforderlichen Tabellen in den entsprechenden Datenbanken

2. Wenn physisch fern, Einrichten der Datenbankserver für die Verwendung der geeigneten Übertragungsprotokolle
3. Wenn physisch fern, Katalogisieren der Knoten und der Datenbanken, um die Datenbanken auf den Datenbankservern zu identifizieren
4. Vorkompilieren Ihres Anwendungsprogramms, um eine Verbindung des Typs 2, d. h. CONNECT 2 im Befehl PRECOMPILE PROGRAM, sowie das einphasige Festschreiben, d. h. SYNCPOINT ONEPHASE im Befehl PRECOMPILE PROGRAM, anzugeben

Wenn sich Datenbanken auf einem Host- oder einem iSeries-Datenbankserver befinden, wird das Produkt DB2 Connect™ zur Herstellung der Konnektivität zu diesen Servern benötigt.

Zugehörige Konzepte:

- „Arbeitseinheiten" auf Seite 31

Zugehörige Tasks:

- „Aktualisieren einer Einzeldatenbank in einer Transaktion" auf Seite 193
- „Aktualisieren mehrerer Datenbanken in einer Transaktion" auf Seite 195

Zugehörige Referenzen:

- „PRECOMPILE Command" in *Command Reference*

Aktualisieren mehrerer Datenbanken in einer Transaktion

Wenn Ihre Daten über mehrere Datenbanken verteilt sind, wollen Sie vielleicht mehrere Datenbanken in einer einzelnen Transaktion lesen und aktualisieren. Diese Art des Datenbankzugriffs wird als *Aktualisierung auf mehreren Systemen* bezeichnet.

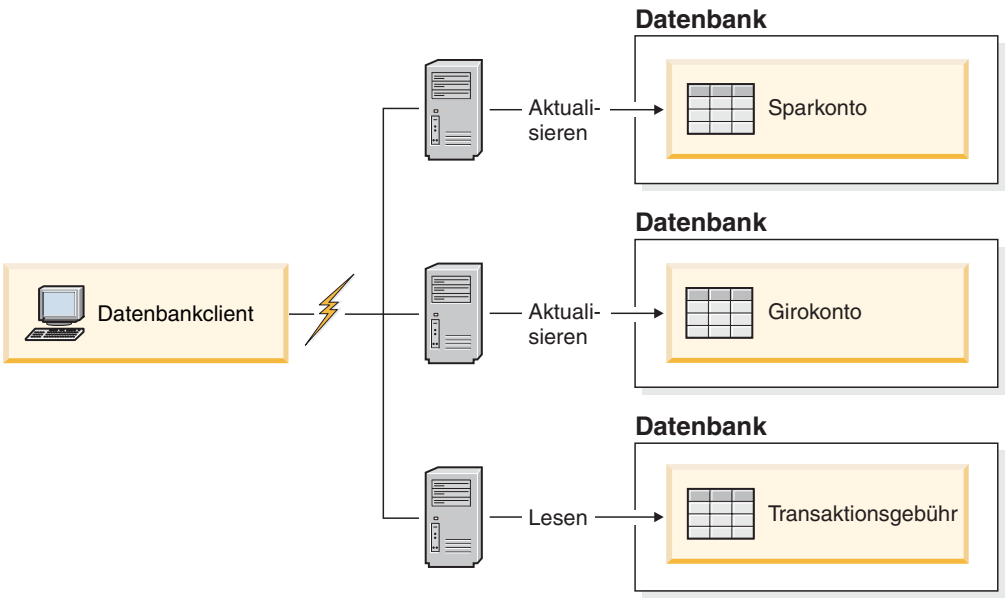


Abbildung 60. Aktualisieren mehrerer Datenbanken in einer einzigen Transaktion

Abb. 60 zeigt einen Datenbankclient, der eine Geldtransaktionsanwendung ausführt, die auf drei Datenbankserver zugreift: einer enthält das Girokonto, ein anderer das Sparkonto und der dritte den Plan für die Transaktionsgebühren.

Vorgehensweise:

Zur Einrichtung einer Geldtransaktionsanwendung für diese Umgebung stehen zwei Optionen zur Auswahl:

1. Unter Verwendung des Transaktionsmanagers (TM) von DB2 Universal Database™ (DB2 UDB):
   a. Erstellen der erforderlichen Tabellen in den entsprechenden Datenbanken
   b. Wenn physisch fern, Einrichten der Datenbankserver für die Verwendung der geeigneten Übertragungsprotokolle
   c. Wenn physisch fern, Katalogisieren der Knoten und der Datenbanken, um die Datenbanken auf den Datenbankservern zu identifizieren
   d. Vorkompilieren Ihres Anwendungsprogramms, um eine Verbindung des Typs 2, d. h. CONNECT 2 im Befehl PRECOMPILE PROGRAM, sowie das zweiphasige Festschreiben, d. h. SYNCPOINT TWOPHASE im Befehl PRECOMPILE PROGRAM, anzugeben
   e. Konfigurieren des DB2 UDB-Transaktionsmanagers (TM)
2. Unter Verwendung eines XA-konformen Transaktionsmanagers:
   a. Erstellen der erforderlichen Tabellen in den entsprechenden Datenbanken
   b. Wenn physisch fern, Einrichten der Datenbankserver für die Verwendung der geeigneten Übertragungsprotokolle
   c. Wenn physisch fern, Katalogisieren der Knoten und der Datenbanken, um die Datenbanken auf den Datenbankservern zu identifizieren
   d. Vorkompilieren Ihres Anwendungsprogramms, um eine Verbindung des Typs 2, d. h. CONNECT 2 im Befehl PRECOMPILE PROGRAM, sowie das einphasige Festschreiben, d. h. SYNCPOINT ONEPHASE im Befehl PRECOMPILE PROGRAM, anzugeben
   e. Konfigurieren des XA-konformen Transaktionsmanagers zur Verwendung der DB2 UDB-Datenbanken

Zugehörige Konzepte:

- „Arbeitseinheiten" auf Seite 31
- „DB2-Transaktionsmanager" auf Seite 196

Zugehörige Tasks:

- „Aktualisieren einer Einzeldatenbank in einer Transaktion" auf Seite 193
- „Aktualisieren einer Einzeldatenbank in einer Transaktion für mehrere Datenbanken" auf Seite 194

Zugehörige Referenzen:

- „PRECOMPILE Command" in *Command Reference*

DB2-Transaktionsmanager

Der Transaktionsmanager (TM) von DB2® Universal Database (DB2 UDB) ordnet Transaktionen Kennungen zu, überwacht die Verarbeitung von Transaktionen und ist zuständig für die Beendigung bzw. für Fehler der Transaktionen. DB2 UDB und DB2 Connect™ stellen einen Transaktionsmanager bereit. Der DB2 UDB-Transaktionsmanager speichert Transaktionsinformationen in der vorgesehenen TM-Datenbank.

Der Datenbankmanager stellt Transaktionsmanagerfunktionen bereit, mit deren Hilfe das Aktualisieren mehrerer Datenbanken innerhalb einer einzelnen Arbeitseinheit koordiniert werden kann. Der Datenbankclient koordiniert automatisch die Arbeitseinheit und verwendet eine *Transaktionsmanagerdatenbank*, um jede Transaktion zu registrieren und ihren Fertigstellungsstatus zu protokollieren.

Sie können den DB2 UDB-Transaktionsmanager mit DB2 UDB-Datenbanken verwenden. Wenn Sie neben den DB2 UDB-Datenbanken über weitere Ressourcen verfügen, die an einer zweiphasigen Festschreibungstransaktion beteiligt sein sollen, können Sie einen dem XA-Standard entsprechenden Transaktionsmanager verwenden.

Zugehörige Konzepte:

- „Arbeitseinheiten" auf Seite 31
- „Konfiguration des Transaktionsmanagers von DB2 Universal Database" auf Seite 197
- „Zweiphasige Festschreibung" auf Seite 200

Konfiguration des Transaktionsmanagers von DB2 Universal Database

Wenn Sie einen XA-konformen Transaktionsmanager wie IBM® WebSphere®, BEA Tuxedo oder Microsoft® Transaction Server verwenden, sollten Sie die Konfigurationsanweisungen für das entsprechende Produkt befolgen.

Wenn Sie zur Koordinierung Ihrer Transaktionen DB2® Universal Database (DB2 UDB) für UNIX®-basierte Systeme oder das Windows®-Betriebssystem verwenden, müssen Sie bestimmte Konfigurationsanforderungen erfüllen. Wenn Sie ausschließlich TCP/IP zur Kommunikation verwenden und DB2 UDB für UNIX, Windows, iSeries™ Version 5, z/OS™ oder OS/390® die einzigen Datenbankserver sind, die an Ihren Transaktionen beteiligt sind, ist die Konfiguration recht einfach.

DB2 Connect™ unterstützt nun keinen SNA-Zugriff auf Host- oder iSeries-Server für zweiphasige Festschreibung mehr.

DB2 Universal Database für UNIX und Windows und DB2 für z/OS, OS/390 und iSeries V5 mit TCP/IP-Konnektivität

Wenn jeder der folgenden Punkte auf Ihre Umgebung zutrifft, gestalten sich die Konfigurationsschritte für eine Aktualisierung auf mehreren Systemen relativ einfach.

- Für die gesamte Kommunikation mit fernen Datenbankservern (einschließlich DB2 UDB für z/OS, OS/390 und iSeries V5) wird ausschließlich TCP/IP verwendet.
- DB2 UDB für UNIX-basierte Systeme, Windows-Betriebssysteme, z/OS, OS/390 oder iSeries V5 sind die einzigen an der Transaktion beteiligten Datenbankserver.

Die Datenbank, die als Transaktionsmanagerdatenbank fungieren soll, wird auf dem Datenbankclient durch den Konfigurationsparameter *tm_database* des Datenbankmanagers bestimmt. Bei der Einstellung dieses Konfigurationsparameters sind folgende Faktoren zu beachten:

- Die Transaktionsmanagerdatenbank kann sein:
  - Eine DB2 UDB für UNIX- oder Windows-Datenbank der Version 8

– Eine DB2 für z/OS- und OS/390-Datenbank der Version 7 oder eine DB2 für OS/390-Datenbank der Versionen 5 oder 6
– Eine DB2 für iSeries V5-Datenbank

DB2 für z/OS-, OS/390- und iSeries V5-Datenbankserver werden als Transaktionsmanagerdatenbanken empfohlen. z/OS-, OS/390- und iSeries V5-Systeme sind in der Regel sicherer als Workstation-Server, so dass sich die Wahrscheinlichkeit unplanmäßiger Ausfälle, Neustarts usw. verringert. Aus diesem Grund sind die Wiederherstellungsprotokolle sicherer, die im Fall einer Resynchronisation verwendet werden.

- Wenn der Wert `1ST_CONN` für den Konfigurationsparameter *tm_database* angegeben ist, wird die erste Datenbank, zu der eine Anwendung eine Verbindung herstellt, als Transaktionsmanagerdatenbank verwendet.

  Bei der Verwendung von `1ST_CONN` ist besondere Sorgfalt geboten. Sie sollten diese Konfiguration nur verwenden, wenn es einfach ist, sicherzustellen, dass alle betroffenen Datenbanken korrekt katalogisiert werden, d. h. wenn sich der Datenbankclient, der die Transaktion einleitet, in demselben Exemplar befindet, das auch die beteiligten Datenbanken, einschließlich der Transaktionsmanagerdatenbank, enthält.

Falls Ihre Anwendung versucht, die Verbindung zu der Datenbank, die als Transaktionsmanagerdatenbank fungiert, zu trennen, wird eine Fehlernachricht ausgegeben. Die Verbindung bleibt in diesem Fall erhalten, bis die Arbeitseinheit festgeschrieben ist.

Konfigurationsparameter

Bei der Einrichtung der Umgebung sind die folgenden Konfigurationsparameter zu berücksichtigen:

Konfigurationsparameter des Datenbankmanagers

- *tm_database*

  Mit diesem Parameter wird der Name der Transaktionsmanagerdatenbank für das jeweilige DB2 UDB-Exemplar definiert.

- *spm_name*

  Mit diesem Parameter wird der Name des Exemplars mit dem DB2 Connect-Synchronisationspunktmanager für den Datenbankmanager definiert. Für eine erfolgreiche Resynchronisation muss der Name innerhalb Ihres gesamten Netzwerks eindeutig sein.

- *resync_interval*

  Mit diesem Parameter wird das Zeitintervall (in Sekunden) angegeben, nach dem der DB2-Transaktionsmanager, der DB2 UDB-Server-Datenbankmanager und der DB2 Connect-Synchronisationspunktmanager den Versuch wiederholen sollen, alle ausstehenden unbestätigten Transaktionen wiederherzustellen.

- *spm_log_file_sz*

  Mit diesem Parameter wird die Größe (in 4-KB-Seiten) der SPM-Protokolldatei angegeben.

- *spm_max_resync*

  Mit diesem Parameter wird die Anzahl der Agenten angegeben, die gleichzeitig Resynchronisationsoperationen durchführen können.

- *spm_log_path*

  Mit diesem Parameter wird der Protokollpfad für die SPM-Protokolldateien angegeben.

Konfigurationsparameter der Datenbank

- *maxappls*

  Mit diesem Parameter wird die zulässige maximale Anzahl aktiver Anwendungen angegeben. Der Wert dieses Parameters muss gleich oder größer sein als die Summe der verbundenen Anwendungen plus die Anzahl dieser Anwendungen, die sich gleichzeitig im Prozess einer zweiphasigen Festschreibung bzw. einer ROLLBACK-Operation befinden können, plus die angenommene Anzahl unbestätigter Transaktionen, die zu einem beliebigen Zeitpunkt gleichzeitig vorhanden sein können.

- *autorestart*

  Mit diesem Konfigurationsparameter der Datenbank wird festgelegt, ob die Routine RESTART DATABASE bei Bedarf automatisch aufgerufen wird. Der Standardwert ist YES (d. h. aktiviert).

  Für eine Datenbank, die unbestätigte Transaktionen enthält, ist eine RESTART DATABASE-Operation für den Neustart erforderlich. Wenn *autorestart* nicht aktiviert ist und die letzte Verbindung zur Datenbank getrennt wurde, schlägt die nächste Verbindungsanforderung fehl, so dass ein explizites Aufrufen des Befehls RESTART DATABASE erforderlich wird. Diese Bedingung bleibt solange bestehen, bis die unbestätigten Transaktionen entweder durch die Resynchronisationsoperation des Transaktionsmanagers oder durch eine vom Administrator eingeleitete heuristische Operation beseitigt wird. Wenn der Befehl RESTART DATABASE abgesetzt wird, wird eine Nachricht zurückgegeben, wenn unbestätigte Transaktionen in der Datenbank vorhanden sind. Der Administrator kann dann mit Hilfe des Befehls LIST INDOUBT TRANSACTIONS und anderer Befehle des Befehlszeilenprozessors Informationen über diese unbestätigten Transaktionen erhalten.

Zugehörige Konzepte:

- „DB2-Transaktionsmanager" auf Seite 196

Zugehörige Tasks:

- „Konfigurieren von IBM TXSeries CICS" auf Seite 228
- „Konfigurieren von IBM TXSeries Encina" auf Seite 228
- „Konfigurieren von BEA Tuxedo" auf Seite 230
- „Konfigurieren von IBM WebSphere Application Server" auf Seite 228

Zugehörige Referenzen:

- „spm_log_path - Protokolldateipfad für SPM" in *Systemverwaltung: Optimierung*
- „autorestart - Automatischer Neustart aktiviert" in *Systemverwaltung: Optimierung*
- „maxappls - Maximale Anzahl aktiver Anwendungen" in *Systemverwaltung: Optimierung*
- „resync_interval - Intervall für Transaktionsresynchronisation" in *Systemverwaltung: Optimierung*
- „tm_database - Name für Transaktionsmanagerdatenbank" in *Systemverwaltung: Optimierung*
- „spm_name - Name des Synchronisationspunktmanagers" in *Systemverwaltung: Optimierung*
- „spm_log_file_sz - Protokolldateigröße für SPM" in *Systemverwaltung: Optimierung*
- „spm_max_resync - SPM-Maximum für Resynchronisationsagenten" in *Systemverwaltung: Optimierung*

Aktualisieren einer Datenbank von einem Host- oder iSeries-Client

Anwendungen, die auf einem Host oder einem iSeries-System ausgeführt werden, können auf Daten zugreifen, die sich auf Datenbankservern unter DB2 Universal Database™ (DB2 UDB) befinden. TCP/IP ist das einzige für diesen Zugriff verwendete Protokoll. DB2 UDB-Server auf allen Plattformen unterstützen keinen SNA-Zugriff von fernen Clients aus mehr.

Vor Version 8 unterstützte der TCP/IP-Zugriff von Host- oder iSeries-Clients nur den Zugriff für einphasige Festschreibung. DB2 UDB ermöglicht jetzt einen TCP/IP-Zugriff für zweiphasige Festschreibung von Host- oder iSeries-Clients aus. Es besteht keine Notwendigkeit, den Synchronisationspunktmanager (SPM) bei der Nutzung des TCP/IP-Zugriffs für zweiphasige Festschreibung zu verwenden.

Die DB2 UDB-TCP/IP-Listenerfunktion muss auf dem Server aktiv sein, auf den durch den Host- oder den iSeries-Client zugegriffen werden soll. Sie können prüfen, ob die TCP/IP-Listenerfunktion aktiv ist, indem Sie mit Hilfe des Befehls 'db2set' feststellen, ob die Registriervariable DB2COMM den Wert „tcpip" hat, und mit Hilfe des Befehls GET DBM CFG feststellen, ob der Konfigurationsparameter **svcename** des Datenbankmanagers auf den Servicenamen gesetzt ist. Wenn die Listenerfunktion nicht aktiv ist, kann sie mit Hilfe der Befehle 'db2set' und UPDATE DBM CFG aktiviert werden.

Zugehörige Referenzen:

- „spm_name - Name des Synchronisationspunktmanagers" in *Systemverwaltung: Optimierung*
- „Kommunikationsvariablen" in *Systemverwaltung: Optimierung*

Zweiphasige Festschreibung

Abb. 61 auf Seite 201 veranschaulicht die Schritte, die zu einer Aktualisierung auf mehreren Systemen gehören. Kenntnisse über die Art und Weise, wie eine Transaktion verwaltet wird, erweisen sich bei der Problemlösung als hilfreich, wenn während des Prozesses der zweiphasigen Festschreibung ein Fehler auftritt.

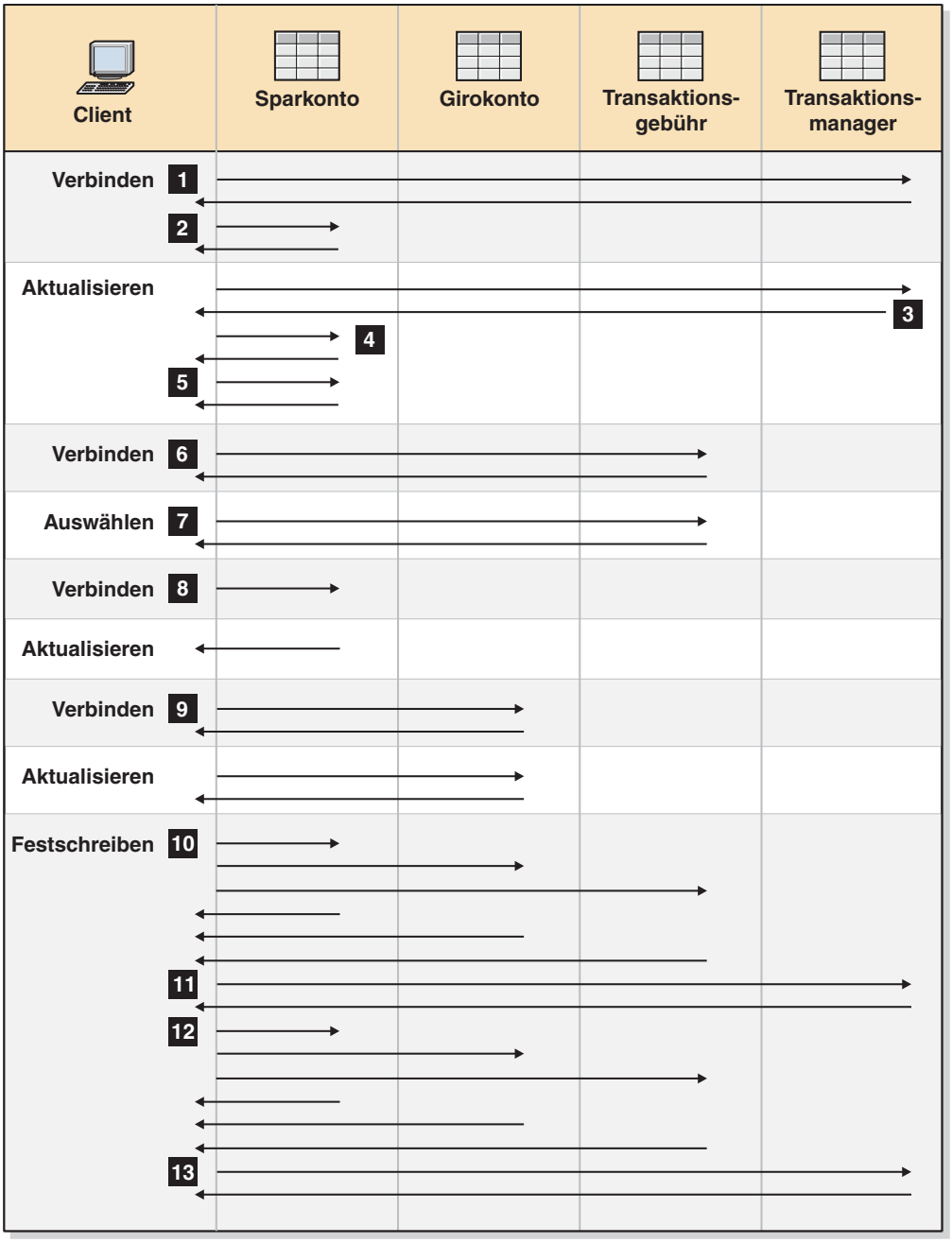


Abbildung 61. Aktualisieren mehrerer Datenbanken

0 Die Anwendung ist für die zweiphasige Festschreibung vorbereitet. Dies kann durch Vorkompilierungsoptionen erreicht werden. Dies kann auch durch eine Konfiguration von DB2® Universal Database (DB2 UDB) CLI (Call Level Interface) erreicht werden.

1 Wenn der Datenbankclient die Verbindung zur Datenbank SPARKO_DB herstellen will, stellt er zunächst intern die Verbindung zur Transaktionsmanagerdatenbank (TMD) her. Die TMD gibt eine Bestätigung an den Datenbankclient zurück. Falls der Konfigurationsparameter *tm_database* des Datenbankmanagers auf den Wert `1ST_CONN` gesetzt ist, wird die Datenbank SPARKO_DB für die Dauer dieses Anwendungsexemplars zur Transaktionsmanagerdatenbank.

2 Die Verbindung zur Datenbank SPARKO_DB wird hergestellt und bestätigt.

3 Der Datenbankclient beginnt mit der Aktualisierung der Tabelle SPAR-_KONTO. Dies ist der Anfang der Arbeitseinheit. Die TMD reagiert auf den Datenbankclient, indem sie eine Transaktions-ID für die Arbeitseinheit bereitstellt. Beachten Sie, dass die Registrierung einer Arbeitseinheit erfolgt, wenn die erste SQL-Anweisung in der Arbeitseinheit ausgeführt wird, und nicht während der Herstellung der Verbindung.

4 Nach dem Empfang der Transaktions-ID registriert der Datenbankclient die Arbeitseinheit bei der Datenbank, die die Tabelle SPAR_KONTO enthält. Der Client erhält eine Antwort, die besagt, dass die Arbeitseinheit erfolgreich registriert wurde.

5 SQL-Anweisungen, die an der Datenbank SPARKO_DB ausgeführt werden, werden auf normale Weise verarbeitet. Die Anwort auf jede Anweisung wird im SQL-Kommunikationsbereich (SQLCA) zurückgegeben, wenn mit SQL-Anweisungen gearbeitet wird, die in ein Programm eingebettet sind.

6 Die Transaktions-ID wird in der Datenbank GEB_DB, die die Tabelle TRANSAKTION_GEB enthält, beim ersten Zugriff auf diese Datenbank innerhalb der Arbeitseinheit registriert.

7 SQL-Anweisungen, die für die Datenbank GEB_DB abgesetzt werden, werden auf normale Weise bearbeitet.

8 Zusätzliche SQL-Anweisungen können für die Datenbank SPARKO_DB ausgeführt werden, indem die Verbindung entsprechend eingerichtet wird. Da die Arbeitseinheit bei der Datenbank SPARKO_DB bereits registriert (**4**) wurde, muss der Datenbankclient den Registrierungsschritt nicht wiederholen.

9 Für die Verbindung mit der Datenbank GIROKO_DB und ihre Verwendung gelten die gleichen Regeln, die unter **6** und **7** beschrieben sind.

10 Wenn der Datenbankclient das Festschreiben der Arbeitseinheit anfordert, wird eine Nachricht *prepare* an alle Datenbanken gesendet, die an der Arbeitseinheit mitwirken. Jede Datenbank schreibt einen Satz "PREPARED" in ihre Protokolldatei und antwortet dem Datenbankclient.

11 Nachdem der Datenbankclient von allen Datenbanken eine positive Antwort erhalten hat, sendet er eine Nachricht an die Transaktionsmanagerdatenbank, die besagt, dass die Arbeitseinheit für das Festschreiben bereit (PREPARED) ist. Die Transaktionsmanagerdatenbank schreibt den Satz "PREPARED" in ihre Protokolldatei und teilt dem Client in einer Antwort mit, dass die zweite Phase der Festschreibung begonnen werden kann.

12 Während der zweiten Phase des Festschreibungsprozesses fordert der Datenbankclient alle beteiligten Datenbanken in einer Nachricht zum Festschreiben auf. Jede Datenbank schreibt den Satz "COMMITTED" in ihre Protokolldatei und gibt die Sperren frei, die für diese Arbeitseinheit aktiv waren. Wenn die Datenbank das Festschreiben der Änderungen beendet hat, sendet sie eine Antwort an den Client.

13 Nachdem der Datenbankclient von allen beteiligten Datenbanken eine positive Antwort erhalten hat, sendet er eine Anforderung an die Transaktionsmanagerdatenbank, um ihr mitzuteilen, dass die Arbeitseinheit beendet wurde. Daraufhin schreibt die Transaktionsmanagerdatenbank einen Satz

"COMMITTED" in ihre Protokolldatei, der anzeigt, dass die Arbeitseinheit beendet ist, und sendet eine Antwort an den Client, dass der Prozess beendet wurde.

Zugehörige Konzepte:

- „Arbeitseinheiten" auf Seite 31
- „DB2-Transaktionsmanager" auf Seite 196

Fehlerbehebung während einer zweiphasigen Festschreibung

Das Beheben von Fehlerbedingungen ist eine normale Aufgabe bei der Anwendungsprogrammierung, Systemverwaltung, Datenbankverwaltung sowie beim Systembetrieb. Durch die Verteilung von Datenbanken auf mehrere ferne Server erhöht sich das Potenzial von Fehlerbedingungen, die aus Netzwerk- oder Übertragungsstörungen resultieren können. Um die Datenintegrität zu gewährleisten, stellt der Datenbankmanager den Prozess der zweiphasigen Festschreibung zur Verfügung. Im Folgenden wird erläutert, wie der Datenbankmanager Fehler während des Prozesses der zweiphasigen Festschreibung behandelt:

- **Fehler in der ersten Phase**
  Wenn eine Datenbank mitteilt, dass sie die Festschreibung der Arbeitseinheit nicht vorbereiten (und kein "PREPARED" in die Protokolldatei eintragen) konnte, macht der Datenbankclient die Arbeitseinheit in der zweiten Phase des Festschreibungsprozesses rückgängig. In diesem Fall wird *keine* PREPARE-Nachricht an die Transaktionsmanagerdatenbank gesendet.

  Während der zweiten Phase der Festschreibung weist der Client alle beteiligten Datenbanken, die während der ersten Phase die Festschreibung erfolgreich vorbereitet haben, an, die Arbeitseinheit rückgängig (ROLLBACK) zu machen. Jede Datenbank schreibt daraufhin einen Satz "ABORT" in ihre Protokolldatei und gibt die Sperren frei, die für diese Arbeitseinheit aktiv waren.
- **Fehler in der zweiten Phase**
  Die Fehlerbehandlung ist in diesem Fall davon abhängig, ob die Transaktion in der zweiten Phase festgeschrieben oder rückgängig gemacht wird. Die zweite Phase macht die Transaktion nur dann rückgängig, wenn in der ersten Phase ein Fehler auftrat.

  Wenn eine der beteiligten Datenbanken die Arbeitseinheit nicht festschreiben kann (u. U. aufgrund eines Übertragungsfehlers), versucht die Transaktionsmanagerdatenbank erneut, für die fehlgeschlagene Datenbank eine Festschreibung auszuführen. Die Anwendung wird jedoch über den SQL-Kommunikationsbereich (SQLCA) darüber informiert, dass die Festschreibung erfolgreich war. DB2® Universal Database (DB2 UDB) stellt sicher, dass die nicht festgeschriebene Transaktion im Datenbankserver festgeschrieben wird. Mit Hilfe des Konfigurationsparameters *resync_interval* des Datenbankmanagers wird das Intervall festgelegt, das die Transaktionsmanagerdatenbank zwischen den einzelnen Versuchen, die Arbeitseinheit festzuschreiben, verstreichen lässt. Alle Sperren bleiben auf dem Datenbankserver aktiv, bis die Arbeitseinheit festgeschrieben wird.

  Fällt die Transaktionsmanagerdatenbank aus, resynchronisiert sie die Arbeitseinheit, wenn sie erneut gestartet wird. Der Resynchronisationsprozess versucht, alle *unbestätigten Transaktionen* zu beenden, d. h. die Transaktionen, die die erste Phase der Festschreibung beendet haben, deren zweite Phase jedoch nicht abgeschlossen wurde. Der Datenbankmanager, unter dem die Transaktionsmanagerdatenbank aktiv ist, führt zur Resynchronisation folgende Schritte aus:

1. Herstellen einer Verbindung zu den Datenbanken, die während der ersten Phase des Festschreibungsprozesses mitteilten, dass sie zur Festschreibung bereit ("PREPARED") waren.
2. Versuch einer Festschreibung der unbestätigten Transaktionen in diesen Datenbanken. (Können die unbestätigten Transaktionen nicht gefunden werden, nimmt der Datenbankmanager an, dass die Datenbank die Transaktionen in der zweiten Phase des Festschreibungsprozesses erfolgreich festgeschrieben hat.)
3. Festschreiben der unbestätigten Transaktionen in der Transaktionsmanagerdatenbank, nachdem alle unbestätigten Transaktionen in den beteiligten Datenbanken festgeschrieben wurden.

Wenn bei einer der beteiligten Datenbanken ein Fehler auftritt und die Datenbank erneut gestartet wird, fragt der Datenbankmanager für diese Datenbank die Transaktionsmanagerdatenbank nach dem Status dieser Transaktion ab, um festzustellen, ob die Transaktion rückgängig gemacht werden sollte. Wenn die Transaktion im Protokoll nicht gefunden wird, nimmt der Datenbankmanager an, dass die betreffende Transaktion rückgängig gemacht wurde, und macht die unbestätigte Transaktion in dieser Datenbank rückgängig. Andernfalls wartet die Datenbank auf eine Festschreibungsanforderung der Transaktionsmanagerdatenbank. Wenn die Transaktion von einem Transaktionsverarbeitungsmonitor (XA-konformem Transaktionsmanager) koordiniert wurde, ist die Datenbank immer davon abhängig, dass die Resynchronisation vom TP-Monitor eingeleitet wird.

Wenn Sie aus einem bestimmten Grund nicht darauf warten können, dass der Transaktionsmanager unbestätigte Transaktionen automatisch auflöst, haben Sie die Möglichkeit, Maßnahmen zur manuellen Auflösung unbestätigter Transaktionen zu ergreifen. Dieser manuelle Prozess wird manchmal als das „Treffen einer heuristischen Entscheidung" bezeichnet.

Fehlerbehebung, wenn autorestart=off

Wenn der Konfigurationsparameter *autorestart* der Datenbank den Wert `OFF` hat und unbestätigte Transaktionen in der Transaktionsmanagerdatenbank oder den Ressourcenmanagerdatenbanken vorhanden sind, muss der Befehl RESTART DATABASE ausgeführt werden, um den Resynchronisationsprozess zu starten. Wenn der Befehl RESTART DATABASE über den Befehlszeilenprozessor ausgeführt wird, verwenden Sie verschiedene Sitzungen. Wenn Sie eine andere Datenbank aus derselben Sitzung erneut starten, wird die Verbindung, die durch die vorherige Ausführung des Befehls RESTART DATABASE hergestellt wurde, beendet und muss erneut gestartet werden. Setzen Sie den Befehl TERMINATE ab, um die Verbindung zu beenden, wenn der Befehl LIST INDOUBT TRANSACTIONS keine weiteren unbestätigten Transaktionen mehr liefert.

Zugehörige Konzepte:

- „Zweiphasige Festschreibung" auf Seite 200

Zugehörige Tasks:

- „Manuelles Auflösen unbestätigter Transaktionen" auf Seite 220

Zugehörige Referenzen:

- „autorestart - Automatischer Neustart aktiviert" in *Systemverwaltung: Optimierung*
- „LIST INDOUBT TRANSACTIONS Command" in *Command Reference*
- „TERMINATE Command" in *Command Reference*
- „RESTART DATABASE Command" in *Command Reference*

Kapitel 7. Entwerfen für XA-konforme Transaktionsmanager

Wenn Sie neben den DB2-Datenbanken über weitere Ressourcen verfügen, die an einer zweiphasigen Festschreibungstransaktion beteiligt sein sollen, kann es sinnvoll sein, einen dem XA-Standard entsprechenden Transaktionsmanager zu verwenden. Wenn Ihre Transaktionen nur auf DB2-Datenbanken zugreifen, sollten Sie den DB2-Transaktionsmanager verwenden, der in „Aktualisieren mehrerer Datenbanken in einer Transaktion" auf Seite 195 beschrieben wird.

Die folgenden Themen enthalten Informationen zur Verwendung des Datenbankmanagers mit einem XA-konformen Transaktionsmanager wie IBM WebSphere oder BEA Tuxedo.

Informationen zu Microsoft Transaction Server finden Sie im Handbuch *CLI Guide and Reference, Volume 1*.

Wenn Sie einen dem XA-Standard entsprechenden Transaktionsmanager verwenden oder implementieren, stehen weitere Informationen auf unserer Website für technische Unterstützung zur Verfügung:

`http://www.ibm.com/software/data/db2/udb/winos2unix/support`

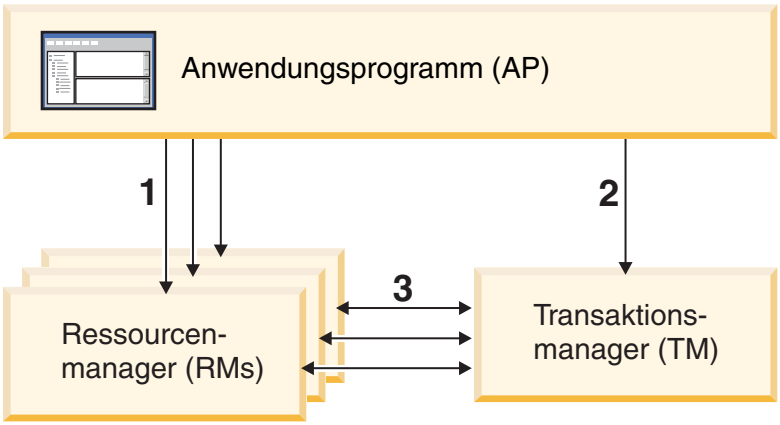
Dort wählen Sie ″DB2 Universal Database″ aus, und durchsuchen anschließend die Website mit dem Schlüsselwort ″XA″ nach den neuesten verfügbaren Informationen zu Transaktionsmanagern, die dem XA-Standard entsprechen.

X/Open-Modell für die verteilte Transaktionsverarbeitung

Das X/Open-Modell für die verteilte Transaktionsverarbeitung (DTP - Distributed Transaction Processing) umfasst drei in Wechselbeziehung zueinander stehende Komponenten:

- Anwendungsprogramm (AP)
- Transaktionsmanager (TM)
- Ressourcenmanager (RM)

Abb. 62 veranschaulicht dieses Modell und zeigt die Beziehungen dieser Komponenten untereinander.



Legende

1 - AP nutzt Ressourcen aus einer Gruppe von RMs
2 - AP definiert Transaktionsgrenzen über TM-Schnittstellen
3 - TM und RMs tauschen Transaktionsinformationen aus

Abbildung 62. X/Open-Modell für die verteilte Transaktionsverarbeitung (DTP)

Anwendungsprogramm (AP)

Das Anwendungsprogramm (AP) definiert die Transaktionsgrenzen und gibt die anwendungsspezifischen Aktionen an, aus denen die Transaktion besteht.

Ein CICS®*-Anwendungsprogramm möchte zum Beispiel auf Ressourcenmanager (RMs) wie beispielsweise eine Datenbank und eine CICS-Warteschlange mit Übergangsdaten (CICS Transient Data Queue) zugreifen und eine Programmlogik verwenden, um die Daten zu bearbeiten. Jede Zugriffsanforderung wird an die entsprechenden Ressourcenmanager durch für diesen Ressourcenmanager spezifische Funktionsaufrufe übergeben. Im Fall von DB2® Universal Database (DB2 UDB) können diese Funktionsaufrufe vom DB2 UDB-Precompiler für jede SQL-Anweisung generiert worden sein, oder es kann sich um Datenbankaufrufe handeln, die direkt vom Programmierer mit Hilfe der APIs codiert wurden.

Zum Transaktionsmanagerprodukt (TM-Produkt) gehört in der Regel ein Transaktionsverarbeitungsmonitor (TP-Monitor) zur Ausführung der Benutzeranwendung. Der TP-Monitor stellt APIs bereit, die einer Anwendung die Möglichkeit geben, eine Transaktion zu starten und zu beenden, und die Funktionen zur Terminierung der zeitlichen Abläufe von Anwendungen sowie zum Lastausgleich unter den vielen Benutzern, die die Anwendung ausführen möchten, zur Verfügung stellen. Das Anwendungsprogramm (AP) in einer DTP-Umgebung ist im Grunde eine Kombination aus der Benutzeranwendung und dem TP-Monitor.

Um eine effiziente Online-Transaktionsverarbeitung (OLTP, Online Transaction Processing) zu ermöglichen, ordnet der TP-Monitor eine Reihe von Server-Prozessen beim Start im Voraus zu und verwaltet und verwendet diese anschließend für die zahlreichen Benutzertransaktionen. Auf diese Weise werden Systemressourcen eingespart, da mehr Benutzer mit Hilfe einer geringeren Anzahl von Server-Prozessen und ihrer zugehörigen RM-Prozesse unterstützt werden. Durch die mehrfache Verwendung dieser Prozesse wird auch der Systemaufwand vermieden, der mit dem Starten eines Prozesses im Transaktionsmanager oder den Ressourcenmanagern (RM) für jede Benutzertransaktion bzw. jedes Benutzerprogramm verbunden ist. (Ein Programm ruft ein oder mehrere Transaktionen auf.) Dies bedeutet auch, dass die Server-Prozesse gegenüber dem Transaktionsmanager und den Ressourcenmanagern als die wirklichen „Benutzerprozesse" auftreten. Hieraus ergeben sich bestimmte Konsequenzen für die Sicherheitsverwaltung und die Anwendungsprogrammierung.

Die folgenden Arten von Transaktionen sind von einem TP-Monitor aus möglich:

- Nicht-XA-Transaktionen

  Diese Transaktionen betreffen Ressourcenmanager, die für den Transaktionsmanager nicht definiert sind und daher nicht unter dem Protokoll für die zweiphasige Festschreibung des Transaktionsmanagers koordiniert werden. Diese Transaktionen können erforderlich werden, wenn eine Anwendung auf einen Ressourcenmanager zugreifen muss, der die XA-Schnittstelle nicht unterstützt. Der TP-Monitor stellt einfach effiziente Einrichtungen zur Zeitplanung und zum Lastausgleich bereit. Da der Transaktionsmanager den Ressourcenmanager nicht explizit für eine XA-Verarbeitung „öffnet", behandelt der Ressourcenmanager diese Anwendung wie jede andere Anwendung, die in einer Nicht-DTP-Umgebung ausgeführt wird.
- Globale Transaktionen

  Diese Transaktionen arbeiten mit Ressourcenmanagern, die gegenüber dem Transaktionsmanager definiert sind und der Steuerung der zweiphasigen Festschreibung des Transaktionsmanagers unterliegen. Eine globale Transaktion stellt eine Arbeitseinheit dar, die auf einen oder mehrere Ressourcenmanager zugreift. Eine *Transaktionsverzweigung* ist der Teil der Arbeit zwischen einem Transaktionsmanager und einem Ressourcenmanager, der die globale Transaktion unterstützt. Eine globale Transaktion kann mehrere Transaktionsverzweigungen umfassen, wenn auf mehrere Ressourcenmanager über einen oder mehrere vom Transaktionsmanager koordinierte Anwendungsprozesse zugegriffen wird.

  Lose gekoppelte globale Transaktionen sind vorhanden, wenn jeder Prozess einer Anzahl von Anwendungsprozessen auf die Ressourcenmanager so zugreift, als wäre er in einer getrennten globalen Transaktion, die Anwendungen jedoch alle der Koordination durch den Transaktionsmanager unterliegen. Jeder Anwendungsprozess verfügt über eine eigene Transaktionsverzweigung innerhalb eines Ressourcenmanagers. Wenn eine COMMIT- oder ROLLBACK-Operation von einem der Anwendungsprogramme, dem Transaktionsmanager oder den Ressourcenmanagern angefordert wird, werden alle Transaktionsverzweigungen zusammen abgeschlossen. Es obliegt den Anwendungen, gegenseitige Sperren von Ressourcen unter den Transaktionsverzweigungen auszuschließen. (Beachten Sie, dass die Koordination der Transaktionen, die durch den DB2 UDB-Transaktionsmanager für Anwendungen implementiert wird, die mit der Option SYNCPOINT(TWOPHASE) vorbereitet (PREPARE) wurden, diesen lose gekoppelten globalen Transaktionen ungefähr entspricht.)

  Fest gekoppelte globale Transaktionen sind vorhanden, wenn mehrere Anwendungsprozesse abwechselnd dieselbe Transaktionsverzweigung in einem Ressourcenmanager verwenden, um Operationen auszuführen. Dem Ressourcen-

manager gegenüber bilden die beiden Anwendungsprozesse eine Einheit. Der Ressourcenmanager muss dafür sorgen, dass keine gegenseitigen Ressourcensperren innerhalb der Transaktionsverzweigung auftreten.

Transaktionsmanager (TM)

Der Transaktionsmanager (TM) ordnet Transaktionen Kennungen zu, überwacht die Verarbeitung von Transaktionen und ist zuständig für die Beendigung bzw. für Fehler der Transaktionen. Die Kennungen für Transaktionsverzweigungen (XIDs) werden vom Transaktionsmanager zugeordnet, um die globale Transaktion und die spezielle Verzweigung innerhalb eines Ressourcenmanagers (RM) zu identifizieren. Diese Kennung ist das Korrelationstoken zwischen dem Protokoll in einem Transaktionsmanager (TM) und dem Protokoll in einem Ressourcenmanager (RM). Die XID wird für eine zweiphasige Festschreibung bzw. für eine ROLLBACK-Operation benötigt, um die *Resynchronisation* (abgekürzt *resync*) bei einem Systemstart durchzuführen oder um dem Administrator die Möglichkeit zu geben, bei Bedarf eine *heuristische* Operation (auch als *manueller Eingriff* bezeichnet) vorzunehmen.

Nach dem Start fordert ein TP-Monitor den Transaktionsmanager auf, alle Ressourcenmanager (RMs) zu öffnen, die in einer Gruppe von Anwendungsservern definiert sind. Der Transaktionsmanager übergibt die **xa_open**-Aufrufe an die Ressourcenmanager, damit diese für die DTP-Verarbeitung initialisiert werden können. Im Rahmen dieser Startprozedur führt der Transaktionsmanager eine Resynchronisation (resync) durch, um alle *unbestätigten Transaktionen* aufzulösen. Eine unbestätigte Transaktion ist eine globale Transaktion, die nicht vollständig zu Ende geführt wurde. Dies geschieht, wenn auf den Transaktionsmanager (oder mindestens auf einen Ressourcenmanager) nach einer erfolgreich abgeschlossenen ersten Phase (Vorbereitunsphase) des Protokolls für zweiphasige Festschreibung nicht mehr zugegriffen werden kann. Der Ressourcenmanager weiß solange nicht, ob seine Verzweigung der Transaktion festzuschreiben oder rückgängig zu machen ist, bis der Transaktionsmanager sein eigenes Protokoll mit den Protokollen der Ressourcenmanager abgleichen kann, wenn sie wieder verfügbar werden. Zur Durchführung der Resynchronisation setzt der Transaktionsmanager einen **xa_recover**-Aufruf an jeden Ressourcenmanager mindestens einmal ab, um alle unbestätigten Transaktionen zu identifizieren. Der Transaktionsmanager vergleicht die Antworten mit den Informationen im eigenen Protokoll, um zu bestimmen, ob die Ressourcenmanager angewiesen werden sollen, eine **xa_commit**-Operation oder eine **xa_rollback**-Operation für diese Transaktionen durchzuführen. Wenn ein Ressourcenmanager seine Verzweigung einer unbestätigten Transaktion aufgrund einer heuristischen Operation durch den Administrator bereits festgeschrieben oder rückgängig gemacht hat, setzt der Transaktionsmanager einen Aufruf **xa_forget** an diesen Ressourcenmanager ab, um die Resynchronisation abzuschließen.

Wenn eine Benutzeranwendung eine COMMIT- oder ROLLBACK-Operation anfordert, muss sie die vom TP-Monitor oder Transaktionsmanager bereitgestellte API verwenden, so dass der Transaktionsmanager die COMMIT- und ROLLBACK-Operation unter allen beteiligten Ressourcenmanagern koordinieren kann. Wenn zum Beispiel eine CICS-Anwendung die CICS-Anforderung SYNCPOINT absetzt, um eine Transaktion festzuschreiben, setzt der Transaktionsmanager CICS XA TM (im Encina® Server implementiert) seinerseits die entsprechenden XA-Aufrufe wie **xa_end**, **xa_prepare**, **xa_commit** oder **xa_rollback** ab, um den Ressourcenmanager anzuweisen, die Transaktion festzuschreiben oder rückgängig zu machen. Der Transaktionsmanager kann eine einphasige Festschreibung anstelle der zweiphasigen Festschreibung durchführen, wenn nur ein Ressourcenmanager beteiligt ist oder wenn ein Ressourcenmanager antwortet, dass seine Transaktionsverzweigung im reinen Lesemodus arbeitet.

Ressourcenmanager (RM)

Ein Ressourcenmanager (RM) stellt den Zugriff auf gemeinsame Ressourcen wie Datenbanken zur Verfügung.

DB2 UDB als Ressourcenmanager einer Datenbank kann an einer *globalen Transaktion* beteiligt sein, die von einem dem XA-Standard entsprechenden Transaktionsmanager koordiniert wird. Wie für die XA-Schnittstelle erforderlich stellt der Datenbankmanager eine externe C-Variable *db2xa_switch* des Typs xa_switch_t bereit, um die XA-Schalterstruktur an den Transaktionsmanager zurückzugeben. Diese Datenstruktur enthält die Adressen der verschiedenen XA-Routinen, die vom Transaktionsmanager aufzurufen sind, und die Betriebsmerkmale des Ressourcenmanagers.

Es gibt zwei Methoden, mit denen der Ressourcenmanager seine Teilnahme an der jeweiligen globalen Transaktion registrieren kann: *statische Registrierung* und *dynamische Registrierung*:

- Für die statische Registrierung ist es erforderlich, dass der Transaktionsmanager (für jede Transaktion) die Folge von Aufrufen **xa_start**, **xa_end** und **xa_prepare** an alle Ressourcenmanager absetzt, die für die Server-Anwendung definiert sind, unabhängig davon, ob der einzelne Ressourcenmanager von der Transaktion verwendet wird. Dies ist nicht effizient, wenn nicht jeder Ressourcenmanager an jeder Transaktion beteiligt ist. Der Grad der Ineffizienz ist proportional zur Anzahl der definierten Ressourcenmanager.
- Die (von DB2 UDB genutzte) dynamische Registrierung ist flexibel und effizient. Ein Ressourcenmanager registriert sich dem Transaktionsmanager gegenüber nur dann mit Hilfe eines **ax_reg**-Aufrufs, wenn der Ressourcenmanager eine Anforderung für seine Ressource empfängt. Beachten Sie, dass es keine Leistungsnachteile bei dieser Methode gibt, wenn lediglich ein Ressourcenmanager definiert ist oder wenn jeder Ressourcenmanager von jeder Transaktion verwendet wird, da die Aufrufe **ax_reg** und **xa_start** ähnliche Pfade im Transaktionsmanager haben.

Eine XA-Schnittstelle stellt eine Zweiwegeübertragung zwischen einem Transaktionsmanager und einem Ressourcenmanager her. Bei der XA-Schnittstelle handelt es sich um eine Systemebenenschnittstelle zwischen den beiden DTP-Softwarekomponenten, und nicht um eine gewöhnliche Schnittstelle für Anwendungsprogramme, für die ein Entwickler Aufrufe codiert. Anwendungsentwickler sollten jedoch mit den durch DTP-Softwarekomponenten bedingten Programmiereinschränkungen vertraut sein.

Obwohl die XA-Schnittstelle unveränderlich ist, kann jeder dem XA-Standard entsprechende Transaktionsmanager einen Ressourcenmanager auf eine eigene, produktspezifische Weise integrieren. Informationen zur Integration Ihres DB2 UDB-Produkts als Ressourcenmanager mit einem bestimmten Transaktionsmanager finden Sie in der Dokumentation des entsprechenden Transaktionsmanagerprodukts.

Zugehörige Konzepte:

- „Sicherheitsüberlegungen für XA-Transaktionsmanager" auf Seite 223
- „Von DB2 Universal Database unterstützte XA-Funktion" auf Seite 225
- „X/Open XA Interface Programming Considerations" in *Application Development Guide: Programming Client Applications*

Zugehörige Tasks:

- „Aktualisieren mehrerer Datenbanken in einer Transaktion" auf Seite 195

Einrichten von Ressourcenmanagern

Jede Datenbank wird als ein separater Ressourcenmanager (RM) beim Transaktionsmanager (TM) definiert, und die Datenbank muss durch eine XA-Zeichenfolge zum Öffnen identifiziert werden.

Bei der Einrichtung einer Datenbank als Ressourcenmanager ist keine XA-Zeichenfolge zum Schließen (xa_close) erforderlich. Falls eine solche Zeichenfolge angegeben wird, wird sie vom Datenbankmanager ignoriert.

Überlegungen zu Datenbankverbindungen

Automatische Clientweiterleitung (ACR)

Immer wenn ein Server ausfällt, empfängt jeder Client, der mit diesem Server verbunden ist, einen Kommunikationsfehler, der die Verbindung beendet und zu einem Anwendungsfehler führt. In Anwendungsumgebungen, in denen Verfügbarkeit eine wichtige Rolle spielt, besitzt der Benutzer entweder eine redundante Konfiguration oder sorgt für eine Übernahme der Serverfunktion durch einen Bereitschaftsknoten. In beiden Fällen versucht der Client-Code von DB2® Universal Database (DB2 UDB) die Verbindung entweder zur ursprünglichen Datenbank (die nun vielleicht auf einem Übernahmeknoten ausgeführt wird, der auch die IP-Adresse übernimmt) oder zu einer neuen Datenbank auf einem anderen Server. Die Anwendung wird dann über einen SQLCODE-Wert benachrichtigt, dass die Verbindung umgeleitet wurde und dass die spezielle Transaktion, die gerade ausgeführt wurde, rückgängig gemacht wurde. An diesem Punkt kann die Anwendung die Transaktion erneut ausführen oder einfach fortfahren.

Die Datenkonsistenz zwischen der ausgefallenen primären Datenbank und der Bereitschaftsdatenbank, welche die Funktion übernimmt, ist bei Verwendung der automatischen Clientweiterleitung (ACR - Automatic Client Reroute) in hohem Maße vom Status der Datenbankprotokolle in der Datenbank abhängig, an die die Verbindung weitergeleitet wird. In dieser Erläuterung sollen diese Datenbank als „Bereitschaftsdatenbank" und der Server, auf dem sich diese Bereitschaftsdatenbank befindet, als „Bereitschaftsserver" bezeichnet werden. Wenn die Bereitschaftsdatenbank zum Zeitpunkt des Ausfalls eine exakte Kopie der ausgefallenen Primärdatenbank ist, sind die Daten in der Bereitschaftsdatenbank konsistent und es treten keine Probleme mit der Datenintegrität auf. Wenn die Bereitschaftsdatenbank jedoch keine exakte Kopie der ausgefallenen Primärdatenbank ist, kann es zu Problemen mit der Datenintegrität infolge inkonsistenter Transaktionsergebnisse bei Transaktionen kommen, die durch XA-Transaktionsmanager vorbereitet, aber noch nicht festgeschrieben wurden. Solche Transaktionen werden als unbestätigte Transaktionen bezeichnet. Der Datenbankadministrator und Anwendungsentwickler, die mit der ACR-Funktion arbeiten, müssen sich des Risikos von Problemen der Datenintegrität bei der Verwendung dieser Funktionalität bewusst sein.

In den folgenden Abschnitten werden die verschiedenen DB2 UDB-Datenbankumgebungen sowie die jeweiligen Risiken von Problemen mit der Datenintegrität beschrieben.

HADR (High Availability Disaster Recovery):

Die HADR-Funktion (High Availability Disaster Recovery) von DB2 UDB kann zur Steuerung der Protokollduplizierstufe zwischen der primären und der sekundären Datenbank verwendet werden, wenn die Anwendung die Konnektivität nach

einem Ausfall der Primärdatenbank zurückerhält. Der Datenbankkonfigurationsparameter, der die Protokollduplizierstufe steuert, heißt *hadr_syncmode*. Für diesen Parameter sind drei Werte möglich:

- SYNC

  Dieser Modus bietet den größten Schutz gegen Transaktionsverlust auf Kosten der längsten Transaktionsantwortzeit unter den drei Modi. Wie der Name dieses Modus erkennen lässt, dient SYNC zur Synchronisation des Schreibens des Transaktionsprotokolls in der Primärdatenbank und der Bereitschaftsdatenbank. Die Synchronisation ist abgeschlossen, wenn die Primärdatenbank die eigenen Protokolldateien geschrieben und von der Bereitschaftsdatenbank die Bestätigung empfangen hat, dass die Protokolle auch in der Bereitschaftsdatenbank geschrieben wurden.

  Bei Einsatz eines XA-Transaktionsmanagers zur Koordinierung der Transaktionen mit DB2 UDB-Ressourcen wird ausdrücklich empfohlen, den Modus SYNC zu verwenden. Der Modus SYNC garantiert die Datenintegrität sowie die Integrität der Transaktionsresynchronisation, wenn ein Client an die sekundäre Datenbank weitergeleitet wird, da diese ein exaktes Replikat der Primärdatenbank ist.

- NEARSYNC

  Dieser Modus bietet einen etwas geringeren Schutz gegen Transaktionsverlust, dafür allerdings im Vergleich zum Modus SYNC auch eine kürzere Antwortzeit. Die Primärdatenbank betrachtet das Schreiben von Protokollen nur dann als erfolgreich, wenn Protokolle in die eigenen Protokolldateien geschrieben wurden und die Datenbank eine Bestätigung von der Sekundärdatenbank empfangen hat, dass die Protokolle ebenfalls in den Hauptspeicher auf der Bereitschaftsdatenbank geschrieben wurden. Wenn die Bereitschaftsdatenbank einen Ausfall hat, bevor sie die Protokolle aus dem Speicher auf die Platte kopieren kann, gehen die Protokolle in der Bereitschaftsdatenbank kurzfristig verloren.

  Angesichts der Möglichkeit, dass Datenbankprotokolle verloren gehen können, und der Tatsache, dass die Bereitschaftsdatenbank kein exaktes Replikat der Primärdatenbank ist, kann die Datenintegrität unter Umständen gefährdet werden. Die Gefährdung ist möglich, wenn die betreffende Transaktion noch unbestätigt ist und die Primärdatenbank ausfällt. Nehmen Sie zum Beispiel an, dass das Transaktionsergebnis eine COMMIT-Operation ist. Wenn der XA-Transaktionsmanager (TM) die nachfolgende XA_COMMIT-Anforderung absetzt, schlägt diese fehl, da die Primärdatenbank ausgefallen ist. Da die XA_COMMIT-Anforderung fehlgeschlagen ist, muss der XA-Transaktionsmanager diese Transaktion in dieser Datenbank bereinigen, indem er eine XA_RECOVER-Anforderung absetzt. Die Bereitschaftsdatenbank antwortet, indem sie eine Liste aller Transaktionen zurückgibt, die noch unbestätigt (INDOUBT) sind. Wenn die Bereitschaftsdatenbank ausfallen würde und erneut gestartet werden müsste, bevor die Datenbankprotokolle „im Speicher" auf die Platte geschrieben und die XA_RECOVER-Anforderung vom XA-Transaktionsmanager abgesetzt worden wären, würde die Bereitschaftsdatenbank die Protokollinformationen über die Transaktion verlieren und könnte sie nicht als Antwort auf die XA_RECOVER-Anforderung zurückgeben. Der XA-Transaktionsmanager nähme in diesem Fall an, dass die Datenbank diese Transaktion festgeschrieben hätte. Tatsächlich ist jedoch die Datenbearbeitung verloren gegangen und es entsteht der Eindruck, dass die Transaktion rückgängig gemacht wurde. Dies führt indes zu einem Problem der Datenintegrität, da alle anderen an dieser Transaktion beteiligten Ressourcen durch den XA-Transaktionsmanager zur Festschreibung (COMMITTED) der Transaktion veranlasst wurden.

Der Modus NEARSYNC stellt einen guten Kompromiss zwischen Datenintegrität und Transaktionsantwortzeit dar, da die Wahrscheinlichkeit, dass sowohl die Primärdatenbank als auch die Bereitschaftsdatenbank ausfallen, gering sein sollte. Jedoch muss sich ein Datenbankadministrator über die Möglichkeit von Problemen mit der Datenintegrität im Klaren sein.

- ASYNC

  Dieser Modus unterliegt der größten Wahrscheinlichkeit von Transaktionsverlust bei Ausfall der Primärdatenbank, bietet im Gegenzug jedoch die kürzeste Transaktionsantwortzeit aller drei Modi. Die Primärdatenbank betrachtet das Schreiben von Protokollen nur dann als erfolgreich, wenn Protokolle in die eigenen Protokolldateien geschrieben und die Protokolle an die TCP-Schicht auf der Hostmaschine der Primärdatenbank gesendet wurden. Die Primärdatenbank wartet keinerlei Bestätigung von der Bereitschaftsdatenbank ab. Die Protokolle können noch auf dem Weg zur Bereitschaftsdatenbank sein, während die Primärdatenbank die entsprechenden Transaktionen bereits als festgeschrieben betrachtet.

  Wenn das gleiche wie das für den Modus NEARSYNC beschriebene Szenario auftritt, ist die Wahrscheinlichkeit des Verlusts von Transaktionsinformationen höher als beim Modus NEARSYNC. Daher ist die Wahrscheinlichkeit von Problemen mit der Datenintegrität höher als bei NEARSYNC und somit natürlich auch bei SYNC.

Partitionierte DB2 UDB ESE-Datenbankumgebungen:

Die Verwendung der automatischen Clientweiterleitung in partitionierten Umgebungen kann ebenfalls zu Problemen der Datenintegrität führen. Wenn die Bereitschaftsdatenbank so definiert ist, dass sie eine andere Partition der gleichen Datenbank ist, können sich aus der Bereinigung unbestätigter Transaktionen in Szenarios, wie sie im obigen Abschnitt HADR zum Modus NEARSYNC beschrieben werden, Probleme der Datenintegrität ergeben. Dies ist möglich, weil die Partitionen die Datenbanktransaktionsprotokolle nicht gemeinsam benutzen. Daher erhält die Bereitschaftsdatenbank (Partition B) keine Kenntnis von unbestätigten Transaktionen, die in der Primärdatenbank (Partition A) vorhanden sind.

Nicht partitionierte DB2 UDB ESE-Datenbankumgebungen:

Die Verwendung der automatischen Clientweiterleitung in nicht partitionierten Umgebungen kann ebenfalls zu Problemen der Datenintegrität führen. Unter der Annahme, dass keine Technologie zur Plattenübernahme wie IBM® AIX® High Availability Cluster Multiprocessor (HACMP), Microsoft® Cluster Service (MSCS) oder Service Guard von HP eingesetzt wird, verfügt die Sekundärdatenbank nicht über die Datenbanktransaktionsprotokolle, die in der Primärdatenbank vorhanden sind, wenn diese ausfällt. Daher können sich aus der Bereinigung unbestätigter Transaktionen in Szenarios, wie sie im obigen Abschnitt HADR zum Modus NEARSYNC beschrieben werden, Probleme der Datenintegrität ergeben.

Transaktionen mit Zugriff auf partitionierte Datenbanken

In einer Umgebung mit partitionierten Datenbanken können Benutzerdaten auf mehrere Datenbankpartitionen verteilt werden. Eine Anwendung, die auf eine solche Datenbank zugreift, stellt die Verbindung her und sendet Anforderungen an eine der Partitionen der Datenbank (den Koordinatorknoten). Verschiedene Anwendungen können die Verbindung zu verschiedenen Datenbankpartitionen herstellen, und dieselbe Anwendung kann verschiedene Datenbankpartitionen für verschiedene Verbindungen auswählen.

Für Transaktionen, die an einer Datenbank in einer Umgebung mit partitionierten Datenbanken ausgeführt werden, muss der gesamte Zugriff über die *gleiche* Datenbankpartition erfolgen. Das bedeutet, dass die gleiche Datenbankpartition vom Beginn der Transaktion bis zu dem Zeitpunkt (einschließlich) verwendet werden muss, zu dem die Transaktion festgeschrieben wird.

Jede Transaktion für die partitionierte Datenbank muss festgeschrieben werden, bevor die Verbindung getrennt wird.

Zugehörige Konzepte:

- „X/Open-Modell für die verteilte Transaktionsverarbeitung" auf Seite 205
- „HADR (High Availability Disaster Recovery) - Übersicht" in *Datenwiederherstellung und hohe Verfügbarkeit Handbuch und Referenz*

Zugehörige Referenzen:

- „xa_open-Zeichenfolgeformate" auf Seite 213

xa_open-Zeichenfolgeformate

xa_open-Zeichenfolgeformat für DB2 Universal Database™ (DB2 UDB), DB2 Connect™ Version 8 FixPak 3 und spätere Versionen

Das Format für die xa_open-Zeichenfolge sieht folgendermaßen aus:

```
parm_id1 = <parm-wert>,parm_id2 = <parm-wert>, ...
```

Es spielt keine Rolle, in welcher Reihenfolge diese Parameter angegeben werden. Die gültigen Werte für *parm_id* werden in der folgenden Tabelle beschrieben.

Tabelle 36. Gültige Werte für parm_id

Parametername	Wert	Verbindlich?	Groß-/Kleinschreibung beachtet?	Standardwert
DB	Aliasname der Datenbank	Ja	Nein	Kein
Der Aliasname der Datenbank, der von der Anwendung für den Zugriff auf die Datenbank verwendet wird.				
UID	Benutzer-ID	Nein	Ja	Kein
Eine Benutzer-ID mit Berechtigung zur Verbindung mit der Datenbank. Erforderlich, wenn ein Kennwort angegeben wird.				
PWD	Kennwort	Nein	Ja	Kein
Ein zur Benutzer-ID gehöriges Kennwort. Erforderlich, wenn eine Benutzer-ID angegeben wird.				
TPM	Name des TP-Monitors	Nein	Nein	Kein
Name des verwendeten TP-Monitors. Die unterstützten Werte finden Sie in der nächsten Tabelle. Dieser Parameter kann angegeben werden, um mehreren TP-Monitoren die Verwendung eines einzigen DB2 UDB-Exemplars zu ermöglichen. Der angegebene Wert überschreibt den im Konfigurationsparameter *tp_mon_name* des Datenbankmanagers definierten Wert.				

Tabelle 36. Gültige Werte für parm_id (Forts.)

Parametername	Wert	Verbindlich?	Groß-/Kleinschreibung beachtet?	Standardwert
AXLIB	Bibliothek, die die Funktionen **ax_reg** und **ax_unreg** des TP-Monitors enthält	Nein	Ja	Kein
Dieser Wert wird von DB2 UDB zum Abrufen der Adressen der erforderlichen Funktionen **ax_reg** und **ax_unreg** verwendet. Er kann genutzt werden, um aufgrund des Parameters TPM angenommene Werte zu überschreiben, oder er kann von TP-Monitoren genutzt werden, die nicht in der Liste für TPM aufgeführt sind. Wenn die Bibliothek unter AIX eine Archivierungsbibliothek ist, muss zusätzlich zum Namen der Bibliothek auch der Name der Teildatei angegeben werden. Zum Beispiel: `AXLIB=/usr/mqm/lib/libmqmax_r.a(libmqmax_r.o).`				
CHAIN_END	xa_end-Verkettungsmarkierung. Gültige Werte: `T`, `F` oder kein Wert.	Nein	Nein	F
Die XA_END-Verkettung ist eine Optimierungsmaßnahme, die von DB2 UDB zur Verringerung von Netzwerkübertragungen genutzt werden kann. Wenn die TP-Monitorumgebung so eingerichtet ist, dass gewährleistet werden kann, dass **xa_prepare** innerhalb desselben Thread oder Prozesses unmittelbar nach einem Aufruf von **xa_end** aufgerufen wird und CHAIN_END aktiviert ist, wird die xa_end-Markierung mit dem Befehl **xa_prepare** verkettet, um so eine Netzwerkübertragung einzusparen. Der Wert `T` bedeutet, dass CHAIN_END aktiviert ist. Der Wert `F` bedeutet, dass CHAIN_END inaktiv ist. Kein Wert bedeutet, dass CHAIN_END aktiviert ist. Dieser Parameter kann dazu dienen, die aus einem angegebenen TPM-Wert abgeleitete Einstellung zu überschreiben.				
SUSPEND_CURSOR	Gibt an, ob Cursor beizubehalten sind, wenn ein Transaktionsthread der Steuerung angehalten wird. Gültige Werte: `T`, `F` oder kein Wert.	Nein	Nein	F
TP-Monitore, die eine Transaktionsverzweigung anhalten, können den angehaltenen Thread oder Prozess für andere Transaktionen wiederverwenden. Wenn SUSPEND_CURSOR inaktiviert ist, werden alle Cursor außer Cursorn mit HOLD-Attributen geschlossen. Bei Wiederaufnahme einer angehaltenen Transaktion muss die Anwendung den Cursor erneut abrufen. Wenn SUSPEND_CURSOR aktiv ist, werden geöffnete Cursor nicht geschlossen und stehen der angehaltenen Transaktion bei Wiederaufnahme zur Verfügung. Der Wert `T` bedeutet, dass SUSPEND_CURSOR aktiviert ist. Der Wert `F` bedeutet, dass SUSPEND_CURSOR inaktiv ist. Kein angegebener Wert bedeutet, dass SUSPEND_CURSOR aktiviert ist. Dieser Parameter kann dazu dienen, die aus einem angegebenen TPM-Wert abgeleitete Einstellung zu überschreiben.				

Tabelle 36. Gültige Werte für parm_id (Forts.)

Parametername	Wert	Verbindlich?	Groß-/Klein-schreibung beachtet?	Standardwert
HOLD_CURSOR	Gibt an, ob Cursor über Festschreibungen von Transaktionen hinweg beibehalten werden. Gültige Werte: `T`, `F` oder kein Wert.	Nein	Nein	F
TP-Monitore verwenden Threads oder Prozesse in der Regel für mehrere Anwendungen wieder. Um sicherzustellen, dass neu geladene Anwendungen keine von einer früheren Anwendung geöffneten Cursor übernehmen, werden Cursor nach einer Festschreibung (COMMIT) geschlossen. Wenn HOLD_CURSORS aktiv ist, werden Cursor mit HOLD-Attributen nicht geschlossen und bleiben über Festschreibungsgrenzen (COMMIT-Grenzen) von Transaktionen bestehen. Bei Verwendung dieser Option muss die globale Transaktion aus dem gleichen Steuerungsthread heraus festgeschrieben oder rückgängig gemacht werden. Wenn HOLD_CURSOR inaktiviert ist, wird das Öffnen von Cursorn mit HOLD-Attributen zurückgewiesen. Der Wert `T` bedeutet, dass HOLD_CURSOR aktiviert ist. Der Wert `F` bedeutet, dass HOLD_CURSOR inaktiv ist. Kein angegebener Wert bedeutet, dass HOLD_CURSOR aktiviert ist. Dieser Parameter kann dazu dienen, die aus einem angegebenen TPM-Wert abgeleitete Einstellung zu überschreiben.				
TOC	Die Entität („Thread of Control"), an die alle DB2 UDB-XA-Verbindungen gebunden werden. Gültige Werte: `T` oder `P`, oder nicht definiert.	Nein	Nein	T (Thread des Betriebssystems)
TOC (Thread of Control) ist die Entität, an die alle DB2 UDB-XA-Verbindungen gebunden werden. Alle DB2 UDB-XA-Verbindungen, die innerhalb einer Entität gebildet werden, müssen eindeutig sein. Das heißt, es können nicht zwei Verbindungen zur selben Datenbank innerhalb der Entität vorhanden sein. Der TOC besitzt zwei Parameter: `T` (Thread des Betriebssystems) und `P` (Prozess des Betriebssystems). Wenn der Wert `T` angegeben wird, sind alle DB2 UDB-XA-Verbindungen, die unter einem bestimmten Betriebssystemthread gebildet werden, nur für diesen Thread eindeutig. DB2 UDB-XA-Verbindungen können nicht von mehreren Threads verwendet werden. Jeder Betriebssystemthread muss einen eigenen Satz von DB2 UDB-XA-Verbindungen bilden. Wenn der Wert `P` definiert wird, sind alle DB2 UDB-XA-Verbindungen für den Betriebssystemprozess eindeutig und alle XA-Verbindungen können von den Betriebssystemthreads gemeinsam verwendet werden.				
SREG	Statische Registrierung. Gültige Werte: `T` oder `F`, oder kein Wert.	Nein	Nein	F
DB2 UDB unterstützt zwei Methoden zur Registrierung einer globalen Transaktion. Die erste Methode ist die dynamische Registrierung (Dynamic Registration), bei der DB2 UDB die **ax_reg**-Funktion des Transaktionsprogramms aufruft, um die Transaktion zu registrieren (siehe AXLIB). Die zweite Methode ist die statische Registrierung (Static Registration), bei der das Transaktionsprogramm die XA-API **xa_start** zur Einleitung einer globalen Transaktion aufruft. Beachten Sie, dass sich die dynamische und statische Registrierung gegenseitig ausschließen.				

Tabelle 36. Gültige Werte für parm_id (Forts.)

Parametername	Wert	Verbindlich?	Groß-/Klein-schreibung beachtet?	Standardwert
CREG	**xa_start**-Verkettungs-markierung. Gültige Werte: `T` oder `F`, oder kein Wert.	Nein	Nein	F
Die **xa_start**-Verkettung ist eine Optimierungsmaßnahme, die von DB2 UDB zur Verringerung von Netzwerkübertragungen genutzt wird. Der Parameter ist nur gültig, wenn der TP-Monitor mit statischer Registrierung arbeitet (siehe SREG). Die TP-Monitorumgebung ist so angelegt, dass sie garantieren kann, dass eine SQL-Anweisung unverzüglich nach dem Aufruf der XA-API **xa_start** aufgerufen wird. Wenn der Parameter CREG auf den Wert `T` gesetzt wird, wird die SQL-Anweisung mit der **xa_start**-Anforderung verkettet, so dass eine Netzwerkübertragung eingespart wird. Dieser Parameter kann dazu dienen, die aus einem angegebenen TPM-Wert abgeleitete Einstellung zu überschreiben.				

TPM- und tp_mon_name-Werte

Der Parameter TPM für die XA-Zeichenfolge zum Öffnen (xa_open) und der Konfigurationsparameter *tp_mon_name* des Datenbankmanagers dienen in DB2 UDB zur Angabe des verwendeten TP-Monitors. Der Wert für *tp_mon_name* gilt für das gesamte DB2 UDB-Exemplar. Der Parameter TPM gilt nur für den speziellen XA-Ressourcenmanager. Der Wert für TPM überschreibt den Parameter *tp_mon_name*. Die folgenden Werte für die Parameter TPM und *tp_mon_name* sind gültig:

Tabelle 37. Gültige Werte für TPM und tp_mon_name

TPM-Wert	TP-Monitor- produkt	Interne Einstellungen

CICS	IBM TxSeries CICS	`AXLIB=libEncServer (für Windows)`
		`=/usr/lpp/encina/lib/libEncServer`
		`(für auf UNIX basierende Systeme)`
		`HOLD_CURSOR=T`
		`CHAIN_END=T`
		`SUSPEND_CURSOR=F`

ENCINA	IBM TxSeries Encina Monitor	`AXLIB=libEncServer (für Windows)`
		`=/usr/lpp/encina/lib/libEncServer`
		`(für auf UNIX basierende Systeme)`
		`HOLD_CURSOR=F`
		`CHAIN_END=T`
		`SUSPEND_CURSOR=F`

Tabelle 37. Gültige Werte für TPM und tp_mon_name (Forts.)

TPM-Wert	TP-Monitor- produkt	Interne Einstellungen
MQ	IBM MQSeries	AXLIB=mqmax (für Windows) =/usr/mqm/lib/libmqmax_r.a (für AIX-Anwendungen mit Threads) =/usr/mqm/lib/libmqmax.a (für AIX-Anwendungen ohne Threads) =/opt/mqm/lib/libmqmax.so (für Solaris) =/opt/mqm/lib/libmqmax_r.sl (für HP-Anwendungen mit Threads) =/opt/mqm/lib/libmqmax.sl (für HP-Anwendungen ohne Threads) =/opt/mqm/lib/libmqmax_r.so (für Linux-Anwendungen mit Threads) =/opt/mqm/lib/libmqmax.so (für Linux-Anwendungen ohne Threads) HOLD_CURSOR=F CHAIN_END=F SUSPEND_CURSOR=F
CB	IBM Component Broker	AXLIB=somtrx1i (für Windows) =libsomtrx1 (für UNIX-basierte Systeme) HOLD_CURSOR=F CHAIN_END=T SUSPEND_CURSOR=F
SF	IBM San Francisco	AXLIB=ibmsfDB2 HOLD_CURSOR=F CHAIN_END=T SUSPEND_CURSOR=F
TUXEDO	BEA Tuxedo	AXLIB=libtux HOLD_CURSOR=F CHAIN_END=F SUSPEND_CURSOR=F
MTS	Microsoft Transaction Server	Es ist nicht nötig, DB2 UDB für MTS zu konfigurieren. MTS wird vom ODBC-Treiber von DB2 UDB automatisch erkannt.
JTA	Java Transaction API	Es ist nicht nötig, DB2 UDB für Enterprise Java Servers (EJS) wie IBM WebSphere zu konfigurieren. Der JDBC-Treiber von DB2 UDB erkennt diese Umgebung automatisch. In diesem Fall wird der TPM-Wert ignoriert.

xa_open-Zeichenfolgeformat für frühere Versionen

Frühere Versionen von DB2 UDB arbeiten mit dem im Folgenden beschriebenen Format der XA-Zeichenfolge zum Öffnen (xa_open). Dieses Format wird aus Gründen der Kompatibilität weiterhin unterstützt. Anwendungen sollten nach Möglichkeit auf das neue Format umgestellt werden.

Jede Datenbank wird als ein separater Ressourcenmanager (RM) beim Transaktionsmanager (TM) definiert, und die Datenbank muss durch eine XA-Zeichenfolge zum Öffnen mit der folgenden Syntax identifiziert werden:

```
"db_alias<,benutzer-id,kennwort>"
```

Der *db_alias* ist erforderlich, um den Aliasnamen der Datenbank anzugeben. Der Aliasname ist derselbe Name wie der Datenbankname, sofern Sie nicht explizit

einen Aliasnamen nach der Erstellung der Datenbank katalogisiert haben. Der Benutzername und das Kennwort sind wahlfrei und werden je nach Methode zur Identifikationsüberprüfung verwendet, um Informationen zur Identifikationsüberprüfung an die Datenbank weiterzugeben.

Beispiele

1. Sie verwenden IBM TxSeries CICS unter Windows NT. Der Dokumentation zu TxSeries ist zu entnehmen, dass Sie den Parameter *tp_mon_name* mit dem Wert `libEncServer:C` konfigurieren müssen. Dies ist immer noch ein akzeptables Format. Allerdings haben Sie mit DB2 UDB oder DB2 Connect™ Version 8 FixPak 3 und späteren Versionen folgende Möglichkeit:
   - Angeben des Wertes `CICS` für den Parameter *tp_mon_name* (empfohlen für dieses Szenario):

     ```
     db2 update dbm cfg using tp_mon_name CICS
     ```

     Für jede Datenbank, die in der Initialisierungszeichenfolge Region—> Resources—> Product—> XAD—> Resource manager für CICS definiert ist, geben Sie Folgendes an:

     ```
     db=dbalias,uid=benutzer-id,pwd=kennwort
     ```
   - Für jede Datenbank, die in der Initialisierungszeichenfolge Region—> Resources—> Product—> XAD—> Resource manager für CICS definiert ist, geben Sie Folgendes an:

     ```
     db=dbalias,uid=benutzer-id,pwd=kennwort,tpm=cics
     ```
2. Sie verwenden IBM MQSeries unter Windows NT. Der Dokumentation zu MQSeries ist zu entnehmen, dass Sie den Parameter *tp_mon_name* mit dem Wert `mqmax` konfigurieren müssen. Dies ist immer noch ein akzeptables Format. Allerdings haben Sie mit DB2 UDB oder DB2 Connect Version 8 FixPak 3 und späteren Versionen folgende Möglichkeit:
   - Angeben des Wertes `MQ` für den Parameter *tp_mon_name* (empfohlen für dieses Szenario):

     ```
     db2 update dbm cfg using tp_mon_name MQ
     ```

     Für jede Datenbank, die in der Initialisierungszeichenfolge Region—> Resources—> Product—> XAD—> Resource manager für CICS definiert ist, geben Sie Folgendes an:

     ```
     uid=benutzer-id,db=dbalias,pwd=kennwort
     ```
   - Für jede Datenbank, die in der Initialisierungszeichenfolge Region—> Resources—> Product—> XAD—> Resource manager für CICS definiert ist, geben Sie Folgendes an:

     ```
     uid=benutzer-id,db=dbalias,pwd=kennwort,tpm=mq
     ```
3. Sie verwenden sowohl IBM TxSeries CICS als auch IBM MQSeries unter Windows NT. Es wird ein einziges DB2 UDB-Exemplar verwendet. In diesem Szenario würden Sie wie folgt konfigurieren:
   a. Für jede Datenbank, die in der Initialisierungszeichenfolge Region—> Resources—> Product—> XAD—> Resource manager für CICS definiert ist, geben Sie Folgendes an:

      ```
      pwd=kennwort,uid=benutzer-id,tpm=cics,db=dbalias
      ```
   b. Für jede Datenbank, die als Ressource in den Eigenschaften des Warteschlangenmanagers definiert ist, geben Sie eine XA-Zeichenfolge zum Öffnen wie folgt an:

      ```
      db=dbalias,uid=benutzer-id,pwd=kennwort,tpm=mq
      ```

4. Sie entwickeln einen eigenen XA-konformen Transaktionsmanager (XA TM) unter Windows NT, und Sie wollen DB2 UDB mitteilen, dass die Bibliothek "meinaxlib" die erforderlichen Funktionen **ax_reg** und **ax_unreg** enthält. Die Bibliothek "meinaxlib" befindet sich in einem in der Anweisung PATH definierten Verzeichnis. Sie haben die folgende Möglichkeit:
   - Angeben des Wertes `meinaxlib` für den Parameter *tp_mon_name*:

     ```
     db2 update dbm cfg using tp_mon_name meinaxlib
     ```

     Und für jede Datenbank, die für den XA TM definiert ist, Angeben einer XA-Zeichenfolge zum Öffnen:

     ```
     db=dbalias,uid=benutzer-id,pwd=kennwort
     ```

   - Für jede Datenbank, die für den XA TM definiert ist, Angeben einer XA-Zeichenfolge zum Öffnen:

     ```
     db=dbalias,uid=benutzer-id,pwd=kennwort,axlib=meinaxlib
     ```

5. Sie entwickeln einen eigenen XA-konformen Transaktionsmanager (XA TM) unter Windows NT, und Sie wollen DB2 UDB mitteilen, dass die Bibliothek "meinaxlib" die erforderlichen Funktionen **ax_reg** und **ax_unreg** enthält. Die Bibliothek "meinaxlib" befindet sich in einem in der Anweisung PATH definierten Verzeichnis. Außerdem wollen Sie die XA-END-Verkettung aktivieren. Sie haben die folgende Möglichkeit:
   - Für jede Datenbank, die für den XA TM definiert ist, Angeben einer XA-Zeichenfolge zum Öffnen:

     ```
     db=dbalias,uid=benutzer-id,pwd=kennwort,axlib=meinaxlib,chain_end=T
     ```

   - Für jede Datenbank, die für den XA TM definiert ist, Angeben einer XA-Zeichenfolge zum Öffnen:

     ```
     db=dbalias,uid=benutzer-id,pwd=kennwort,axlib=meinaxlib,chain_end
     ```

Zugehörige Konzepte:

- „X/Open-Modell für die verteilte Transaktionsverarbeitung" auf Seite 205

Zugehörige Referenzen:

- „tp_mon_name - Name des Transaktionsprozessormonitors" in *Systemverwaltung: Optimierung*

Aktualisieren von Host- oder iSeries-Datenbankservern mit einem XA-konformen Transaktionsmanager

Host- und iSeries-Datenbankserver können aktualisierbar sein, je nach Architektur des XA-Transaktionsmanagers.

Vorgehensweise:

Zur Unterstützung von Festschreibungssequenzen aus verschiedenen Prozessen muss der DB2 Connect™-Verbindungskonzentrator aktiviert werden. Zur Aktivierung des DB2 Connect-Verbindungskonzentrators setzen Sie den Konfigurationsparameter *max_connections* des Datenbankmanagers auf einen Wert, der größer ist als der Wert des Parameters *maxagents*. Beachten Sie, dass für den DB2 Connect-Verbindungskonzentrator ein Client mit DB2 Universal Database™ (DB2 UDB) Version 7.1 oder einer späteren Version zur Unterstützung von XA-Festschreibungssequenzen aus verschiedenen Prozessen erforderlich ist.

Darüber hinaus benötigen Sie DB2 Connect mit konfiguriertem DB2 UDB-Synchronisationspunktmanager (SPM).

Zugehörige Referenzen:

- „maxagents - Maximale Anzahl von Agenten" in *Systemverwaltung: Optimierung*
- „max_connections - Maximale Anzahl von Clientverbindungen" in *Systemverwaltung: Optimierung*

Manuelles Auflösen unbestätigter Transaktionen

Ein dem XA-Standard entsprechender Transaktionsmanager (Transaktionsverarbeitungsmonitor - TP-Monitor) arbeitet mit einem ähnlichen zweiphasigen Festschreibeprozess wie der Transaktionsmanager von DB2 Universal Database™ (DB2 UDB). Der Hauptunterschied zwischen diesen beiden Umgebungen besteht darin, dass hier der TP-Monitor die Funktionen zur Protokollierung und Steuerung zur Verfügung stellt, und nicht der DB2-Transaktionsmanager und die Transaktionsmanagerdatenbank.

Bei der Verwendung eines dem XA-Standard entsprechenden Transaktionsmanagers können Fehler auftreten, die denen für den DB2 UDB-Transaktionsmanager ähnlich sind. Ähnlich wie der DB2 UDB-Transaktionsmanager versucht auch ein XA-Transaktionsmanager, unbestätigte Transaktionen zu resynchronisieren.

Wenn Sie aus einem bestimmten Grund nicht darauf warten können, dass der Transaktionsmanager unbestätigte Transaktionen automatisch auflöst, haben Sie die Möglichkeit, Maßnahmen zur manuellen Auflösung unbestätigter Transaktionen zu ergreifen. Dieser manuelle Prozess wird manchmal als das „Treffen einer heuristischen Entscheidung" bezeichnet.

Der Befehl LIST INDOUBT TRANSACTIONS (mit der Option WITH PROMPTING) bzw. eine entsprechende Gruppe von APIs ermöglicht es, unbestätigte Transaktionen abzufragen, festzuschreiben (COMMIT) oder rückgängig zu machen (ROLLBACK). Darüber hinaus können mit diesem Befehl Transaktionen ignoriert werden („FORGET"), die heuristisch festgeschrieben oder rückgängig gemacht wurden, indem die Protokollsätze entfernt und der Protokollspeicherbereich freigegeben werden.

Einschränkungen:

Verwenden Sie diese Befehle (bzw. die entsprechenden APIs) nur mit *äußerster Vorsicht* und nur als allerletzte Möglichkeit. Die beste Strategie ist, zu warten, bis der Transaktionsmanager den Prozess der Resynchronisation durchgeführt hat. Es kann zu Problemen mit der Datenintegrität kommen, wenn Sie eine Transaktion in einer der beteiligten Datenbanken manuell festschreiben oder rückgängig machen und die entgegengesetzte Maßnahme für eine andere beteiligte Datenbank durchgeführt wird. Die Behebung von Problemen der Datenintegrität setzt voraus, dass Sie die Logik der Anwendung und die geänderten oder rückgängig gemachten Daten kennen, und Sie dann eine Wiederherstellung bis zu einem bestimmten Zeitpunkt durchführen bzw. die Datenbankänderungen manuell rückgängig machen oder erneut anwenden.

Wenn Sie nicht darauf warten können, bis der Transaktionsmanager den Prozess zur Resynchronisation einleitet, und die Ressourcen, die von einer unbestätigten Transaktion blockiert werden, freigeben müssen, sind heuristische Maßnahmen

erforderlich. Diese Situation kann auftreten, wenn der Transaktionsmanager für einen längeren Zeitraum zur Durchführung der Resynchronisation nicht verfügbar ist und die unbestätigte Transaktion Ressourcen bindet, die dringend benötigt werden. Eine unbestätigte Transaktion bindet Ressourcen, die dieser Transaktion zugeordnet wurden, als der Transaktionsmanager oder die Ressourcenmanager noch verfügbar waren. Im Fall des Datenbankmanagers gehören zu diesen Ressourcen zum Beispiel die Sperren für Tabellen und Indizes, der Protokollspeicherbereich und der Speicher, der von der Transaktion belegt wird. Jede unbestätigte Transaktion verringert außerdem die maximale Anzahl gleichzeitiger Transaktionen (um den Wert 1), die von der Datenbank verarbeitet werden können. Darüber hinaus kann auch eine Offlinesicherung erst durchgeführt werden, wenn alle unbestätigten Transaktionen aufgelöst wurden.

Die heuristische Funktion FORGET wird in folgenden Situationen benötigt:

- Wenn eine heuristisch festgeschriebene oder rückgängig gemachte Transaktion zu der Bedingung führt, dass das Protokoll voll ist, die in der Ausgabe des Befehls LIST INDOUBT TRANSACTIONS angezeigt wird.
- Wenn eine Offlinesicherung durchgeführt werden soll.

Die heuristische Funktion FORGET gibt den Protokollspeicherbereich frei, der durch eine unbestätigte Transaktion belegt wird. Dies könnte zur Folge haben, dass, wenn ein Transaktionsmanager schließlich einen Resynchronisationsprozess für diese unbestätigte Transaktion durchführt, er eine falsche Entscheidung treffen könnte, andere Ressourcenmanager festzuschreiben oder rückgängig zu machen, weil für die Transaktion in diesem Ressourcenmanager kein Protokollsatz mehr vorhanden wäre. Allgemein impliziert ein „fehlender" Protokollsatz, dass der Ressourcenmanager die Transaktion rückgängig gemacht hat.

Vorgehensweise:

Obwohl es keine absolut sichere Methode zur Durchführung heuristischer Maßnahmen gibt, werden im Folgenden einige allgemeine Richtlinien beschrieben:

1. Stellen Sie die Verbindung zu der Datenbank her, für die alle Transaktionen abgeschlossen werden sollen.
2. Verwenden Sie den Befehl LIST INDOUBT TRANSACTIONS, um die unbestätigten Transaktionen anzuzeigen. Die *xid* stellt die ID der globalen Transaktion dar und ist mit der *xid* identisch, die vom Transaktionsmanager und von anderen Ressourcenmanagern, die an der Transaktion beteiligt sind, verwendet wird.
3. Bestimmen Sie mit Hilfe Ihrer Kenntnisse über die Anwendung und die Betriebsumgebung für jede unbestätigte Transaktion die anderen beteiligten Ressourcenmanager.
4. Stellen Sie fest, ob der Transaktionsmanager verfügbar ist:
   - Wenn der Transaktionsmanager verfügbar ist und die unbestätigte Transaktion in einem Ressourcenmanager dadurch verursacht wurde, dass der Ressourcenmanager in der zweiten Phase der Festschreibung nicht verfügbar war, oder für einen früheren Resynchronisationsprozess, sollten Sie das Protokoll des Transaktionsmanagers überprüfen, um festzustellen, welche Aktion für die anderen Ressourcenmanager durchgeführt wurde. Anschließend sollten Sie dieselbe Aktion für die Datenbank durchführen, d. h. mit dem Befehl LIST INDOUBT TRANSACTIONS die Transaktion entweder heuristisch festschreiben (COMMIT) oder heuristisch rückgängig machen (ROLLBACK).

- Wenn der Transaktionsmanager *nicht* verfügbar ist, müssen Sie den Status der Transaktion in den anderen beteiligten Ressourcenmanagern verwenden, um festzustellen, welche Aktion durchzuführen ist:
  - Wenn mindestens einer der anderen Ressourcenmanager die Transaktion festgeschrieben hat, dann sollten Sie die Transaktion in allen anderen Ressourcenmanagern heuristisch festschreiben (COMMIT).
  - Wenn mindestens einer der anderen Ressourcenmanager die Transaktion rückgängig gemacht hat, sollten Sie die Transaktion heuristisch rückgängig machen (ROLLBACK).
  - Wenn die Transaktion den Status ″prepared″ (unbestätigt) in allen beteiligten Ressourcenmanagern hat, sollten Sie die Transaktion heuristisch rückgängig machen.
  - Wenn einer oder mehrere der anderen Ressourcenmanager nicht verfügbar sind, sollten Sie die Transaktion heuristisch rückgängig machen.

Zum Abfragen von Informationen über unbestätigte Transaktionen aus DB2 UDB auf UNIX- oder Windows-Systemen, stellen Sie eine Verbindung zu der Datenbank her und setzen den Befehl LIST INDOUBT TRANSACTIONS WITH PROMPTING ab bzw. verwenden die entsprechende API.

Zum Abrufen von Informationen zu unbestätigten Transaktionen in Bezug auf Host- oder iSeries-Datenbankserver stehen zwei Möglichkeiten zur Verfügung:

- Sie können Informationen zu unbestätigten Transaktionen direkt vom Host- oder iSeries-Server abrufen.

  Zum Abrufen von Informationen zu unbestätigten Transaktionen direkt aus DB2 für z/OS und OS/390 rufen Sie den Befehl DISPLAY THREAD TYPE(INDOUBT) auf. Mit Hilfe des Befehls RECOVER können Sie eine heuristische Entscheidung treffen. Zum direkten Abrufen von Informationen zu unbestätigten Transaktionen aus DB2 für iSeries rufen Sie den Befehl **wrkcmtdfn** auf.
- Sie können Informationen zu unbestätigten Transaktionen direkt aus dem DB2 Connect-Server abrufen, der für den Zugriff auf den Host- oder iSeries-Datenbankserver verwendet wird.

  Zum Abrufen von Informationen zu unbestätigten Transaktionen vom DB2 Connect-Server stellen Sie zunächst eine Verbindung zum DB2 UDB-Synchronisationspunktmanager her, indem Sie eine Verbindung zu dem DB2 UDB-Exemplar herstellen, das durch den Wert des Konfigurationsparameters *spm_name* des Datenbankmanagers definiert ist. Anschließend setzen Sie den Befehl LIST DRDA INDOUBT TRANSACTIONS WITH PROMPTING ab, um unbestätigte Transaktionen anzuzeigen und heuristische Entscheidungen zu treffen.

Zugehörige Konzepte:

- „Zweiphasige Festschreibung” auf Seite 200

Zugehörige Referenzen:

- „LIST INDOUBT TRANSACTIONS Command” in *Command Reference*
- „LIST DRDA INDOUBT TRANSACTIONS Command” in *Command Reference*

Zugehörige Beispiele:

- „dbxamon.c -- Show and roll back indoubt transactions.”

Sicherheitsüberlegungen für XA-Transaktionsmanager

Der TP-Monitor ordnet eine Gruppe von Server-Prozessen im Voraus zu und führt die Transaktionen von verschiedenen Benutzern unter den IDs der Serverprozesse aus. Für die Datenbank erscheint jeder Serverprozess wie eine große Anwendung, die viele Arbeitseinheiten vereinigt, die alle unter derselben, dem Serverprozess zugeordneten ID ausgeführt werden.

Wenn zum Beispiel in einer AIX®-Umgebung mit CICS® eine TXSeries® CICS-Region gestartet wird, wird ihr der AIX-Benutzername zugeordnet, unter dem sie definiert ist. Alle CICS-Anwendungsserverprozesse werden ebenfalls unter dieser TXSeries CICS-Haupt-ID ausgeführt, die in der Regel als "cics" definiert ist. CICS-Benutzer können CICS-Transaktionen unter ihrer DCE-Anmelde-ID aufrufen und, während sie in CICS arbeiten, ihre ID mit Hilfe der CESN-Anmeldetransaktion auch ändern. In beiden Fällen ist für den Ressourcenmanager die Endbenutzer-ID nicht verfügbar. Folglich kann ein CICS-Anwendungsprozess Transaktionen für viele Benutzer ausführen, aber sie erscheinen dem Ressourcenmanager wie ein einziges Programm mit vielen Arbeitseinheiten von derselben ID "cics". Wahlfrei können Sie eine Benutzer-ID und ein Kennwort in der XA-Zeichenfolge zum Öffnen angeben, so dass diese Benutzer-ID anstelle der ID "cics" für die Verbindung mit der Datenbank verwendet wird.

Die Auswirkungen sind für statische SQL-Anweisungen nicht groß, da zum Zugriff auf die Datenbank die Zugriffsrechte der Person, die das Paket gebunden hat, und nicht die Zugriffsrechte des Endbenutzers verwendet werden. Das bedeutet aber, dass das Zugriffsrecht EXECUTE der Datenbankpakete der Server-ID, und nicht der Endbenutzer-ID erteilt werden muss.

Für dynamische Anweisungen, für deren Zugriff die Authentifizierung zur Laufzeit erfolgt, müssen die Zugriffsrechte der Datenbankobjekte der Server-ID, und nicht dem tatsächlichen Benutzer dieser Objekte erteilt werden. Anstatt sich bei der Steuerung des Zugriffs bestimmter Benutzer auf die Datenbank zu stützen, müssen Sie über das TP-Monitorsystem festlegen, welche Benutzer welche Programme ausführen können. Der Server-ID müssen alle Zugriffsrechte erteilt werden, die für die SQL-Benutzer erforderlich sind.

Um festzustellen, wer auf eine Datenbanktabelle oder -sicht zugegriffen hat, können Sie folgende Schritte unternehmen:

1. Erstellen Sie sich aus der Katalogsicht SYSCAT.PACKAGEDEP eine Liste aller Pakete, die von der Tabelle oder Sicht abhängig sind.
2. Bestimmen Sie die Namen der Server-Programme (z. B. CICS-Programme), die diesen Paketen aufgrund der in Ihrer Installation verwendeten Namenskonvention entsprechen.
3. Bestimmen Sie die Clientprogramme (z. B. CICS-Transaktions-IDs), die diese Programme aufrufen konnten, und verwenden Sie anschließend das Protokoll des TP-Monitors (z. B. das CICS-Protokoll), um festzustellen, wer diese Transaktionen oder Programme wann ausgeführt hat.

Zugehörige Konzepte:

• „X/Open-Modell für die verteilte Transaktionsverarbeitung" auf Seite 205

Konfigurationsüberlegungen für XA-Transaktionsmanager

Bei der Einrichtung der TP-Monitorumgebung sollten die folgenden Konfigurationsparameter berücksichtigt werden:

- *tp_mon_name*

  Mit diesem Konfigurationsparameter des Datenbankmanagers wird der Name des verwendeten TP-Monitorprodukts (z. B. ″CICS″ oder ″ENCINA″) angegeben.

- *tpname*

  Mit diesem Konfigurationsparameter des Datenbankmanagers wird der Name des fernen Transaktionsprogramms angegeben, das der Datenbankclient verwenden muss, wenn er eine Zuordnungsanforderung an den Datenbankserver über das APPC-Übertragungsprotokoll absetzt. Der Wert wird in der Konfigurationsdatei auf dem Server definiert und muss mit dem im SNA-Transaktionsprogramm konfigurierten TP-Namen für den Transaktionsprozessor übereinstimmen.

- *tm_database*

  Da DB2® Universal Database (DB2 UDB) Transaktionen in der XA-Umgebung *nicht* koordiniert, wird dieser Konfigurationsparameter des Datenbankmanagers für XA-koordinierte Transaktionen nicht verwendet.

- *maxappls*

  Mit diesem Konfigurationsparameter der Datenbank wird die zulässige maximale Anzahl aktiver Anwendungen angegeben. Der Wert dieses Parameters muss gleich oder größer sein als die Summe der verbundenen Anwendungen plus die Anzahl dieser Anwendungen, die sich gleichzeitig im Prozess einer zweiphasigen Festschreibung bzw. einer ROLLBACK-Operation befinden können. Diese Summe sollte dann um die vorab geschätzte Anzahl unbestätigter Transaktionen erhöht werden, die zu einem beliebigen Zeitpunkt gleichzeitig vorhanden sein könnten.

  Für eine TP-Monitorumgebung (z. B. TXSeries® CICS®) müssen Sie den Wert des Parameters *maxappls* eventuell erhöhen. Dies würde bei der Sicherstellung helfen, dass alle TP-Monitorprozesse verarbeitet werden können.

- *autorestart*

  Mit diesem Konfigurationsparameter der Datenbank wird festgelegt, ob die Routine RESTART DATABASE bei Bedarf automatisch aufgerufen wird. Der Standardwert ist `YES` (d. h. aktiviert).

  Für eine Datenbank, die unbestätigte Transaktionen enthält, ist eine RESTART DATABASE-Operation für den Neustart erforderlich. Wenn *autorestart* nicht aktiviert ist und die letzte Verbindung zur Datenbank getrennt wurde, schlägt die nächste Verbindungsanforderung fehl, so dass ein explizites Aufrufen des Befehls RESTART DATABASE erforderlich wird. Diese Bedingung bleibt solange bestehen, bis die unbestätigten Transaktionen entweder durch die Resynchronisationsoperation des Transaktionsmanagers oder durch eine vom Administrator eingeleitete heuristische Operation beseitigt wird. Wenn der Befehl RESTART DATABASE abgesetzt wird, wird eine Nachricht zurückgegeben, wenn unbestätigte Transaktionen in der Datenbank vorhanden sind. Der Administrator kann dann mit Hilfe des Befehls LIST INDOUBT TRANSACTIONS und anderer Befehle des Befehlszeilenprozessors Informationen über diese unbestätigten Transaktionen erhalten.

Zugehörige Konzepte:

- „X/Open-Modell für die verteilte Transaktionsverarbeitung” auf Seite 205

Zugehörige Referenzen:

- „autorestart - Automatischer Neustart aktiviert" in *Systemverwaltung: Optimierung*
- „tpname - APPC-Transaktionsprogrammname" in *Systemverwaltung: Optimierung*
- „maxappls - Maximale Anzahl aktiver Anwendungen" in *Systemverwaltung: Optimierung*
- „tm_database - Name für Transaktionsmanagerdatenbank" in *Systemverwaltung: Optimierung*
- „tp_mon_name - Name des Transaktionsprozessormonitors" in *Systemverwaltung: Optimierung*
- „LIST INDOUBT TRANSACTIONS Command" in *Command Reference*
- „RESTART DATABASE Command" in *Command Reference*

Von DB2 Universal Database unterstützte XA-Funktion

DB2® Universal Database (DB2 UDB) unterstützt die Spezifikation XA91, die in *X/Open CAE Specification Distributed Transaction Processing: The XA Specification* definiert ist, mit folgenden Ausnahmen:

- Asynchrone Services

  Die XA-Spezifikation ermöglicht der Schnittstelle die Verwendung asynchroner Services, so dass das Ergebnis einer Anforderung zu einem späteren Zeitpunkt überprüft werden kann. Für den Datenbankmanager müssen die Anforderungen im synchronen Modus aufgerufen werden.
- Statische Registrierung

  Die XA-Schnittstelle ermöglicht zwei Methoden zur Registrierung eines Ressourcenmanagers: statische und dynamische Registrierung. DB2 UDB unterstützt nur die dynamische Registrierung, die ausgereifter und effizienter ist.
- Migration von Zuordnungen

  DB2 UDB unterstützt die Transaktionsmigration zwischen Threads der Steuerung nicht.

Verwendung und Position von XA-Schaltern

Wie für die XA-Schnittstelle erforderlich stellt der Datenbankmanager eine externe C-Variable *db2xa_switch* des Typs xa_switch_t bereit, um die XA-Schalterstruktur an den Transaktionsmanager zurückzugeben. Neben den Adressen verschiedener XA-Funktionen werden folgende Felder zurückgegeben:

Feld	Wert
name	Der Produktname des Datenbankmanagers. Zum Beispiel: DB2 für AIX®.
flags	TMREGISTER | TMNOMIGRATE Gibt explizit an, dass DB2 UDB die dynamische Registrierung verwendet und dass der TM keine Migration von Zuordnungen verwenden soll. Gibt implizit an, dass kein asynchroner Betrieb unterstützt wird.
version	Muss null sein.

Verwenden des XA-Schalters von DB2 Universal Database

Die XA-Architektur verlangt, dass ein Ressourcenmanager (RM) einen *Schalter* bereitstellt, der dem XA-Transaktionsmanager (TM) Zugriff auf die **xa**_-Routinen des Ressourcenmanagers gibt. Ein RM-Schalter verwendet eine Struktur, die als

xa_switch_t bezeichnet wird. Der Schalter enthält den Namen des RMs, Nicht-NULL-Zeiger auf die XA-Eingangspunkte des RMs, eine Markierung (Flag) und eine Versionsnummer.

UNIX-basierte Systeme

Der DB2 UDB-Schalter kann durch eine der folgenden Methoden abgerufen werden:

- Über eine weitere Zwischenstufe. In einem C-Programm kann dies durch Definieren des Makros:

  ```
  #define db2xa_switch (*db2xa_switch)
  ```

  vor Verwendung von *db2xa_switch* erreicht werden.
- Durch Aufrufen von **db2xacic**

  DB2 UDB stellt diese API zur Verfügung, die die Adresse der *db2xa_switch*-Struktur liefert. Der Prototyp dieser Funktion lautet:

  ```
  struct xa_switch_t * SQL_API_FN  db2xacic( )
  ```

Bei beiden Methoden müssen Sie die Anwendung mit `libdb2` verbinden (″linken″).

Windows NT

Der Zeiger auf die *xa_switch*-Struktur, *db2xa_switch*, wird in Form von DLL-Daten exportiert (d. h. bereitgestellt). Dies heißt für eine Anwendung unter Windows® NT, die diese Struktur verwendet, dass sie auf die Struktur mit Hilfe einer von drei Methoden zugreifen muss:

- Über eine weitere Zwischenstufe. In einem C-Programm kann dies durch Definieren des Makros:

  ```
  #define db2xa_switch (*db2xa_switch)
  ```

  vor Verwendung von *db2xa_switch* erreicht werden.
- Bei Verwendung des Microsoft® Visual C++-Compilers kann *db2xa_switch* folgendermaßen definiert werden:

  ```
  extern __declspec(dllimport) struct xa_switch_t db2xa_switch
  ```
- Durch Aufrufen von **db2xacic**

  DB2 UDB stellt diese API zur Verfügung, die die Adresse der *db2xa_switch*-Struktur liefert. Der Prototyp dieser Funktion lautet:

  ```
  struct xa_switch_t * SQL_API_FN  db2xacic( )
  ```

Bei jeder dieser Methoden müssen Sie die Anwendung mit `db2api.lib` verbinden (″linken″).

C-Beispielcode

Der folgende Code veranschaulicht die verschiedenen Methoden des Zugriffs auf *db2xa_switch* über ein C-Programm auf einer beliebigen DB2 UDB-Plattform. Stellen Sie sicher, dass die Anwendung mit der entsprechenden Bibliothek verbunden wird.

```
#include <stdio.h>
#include <xa.h>

struct xa_switch_t * SQL_API_FN  db2xacic( );

#ifdef DECLSPEC_DEFN
extern __declspec(dllimport) struct xa_switch_t db2xa_switch;
```

```
   #else
   #define db2xa_switch (*db2xa_switch)
   extern struct xa_switch_t db2xa_switch;
   #endif
main( )
   {
      struct xa_switch_t *foo;
      printf ( "%s \n", db2xa_switch.name );
      foo = db2xacic();
      printf ( "%s \n", foo->name );
      return ;
   }
```

Zugehörige Konzepte:

- „X/Open-Modell für die verteilte Transaktionsverarbeitung” auf Seite 205

Fehlerbestimmung für die XA-Schnittstelle

Wenn ein Fehler während einer XA-Anforderung vom Transaktionsmanager festgestellt wird, ist das Anwendungsprogramm eventuell nicht in der Lage, den Fehlercode vom Transaktionsmanager zu ermitteln. Wenn Ihr Programm abnormal beendet wird oder einen unklaren Rückkehrcode vom TP-Monitor oder Transaktionsmanager empfängt, sollten Sie das Serviceprotokoll des DB2-Diagnoseprogramms überprüfen, in dem XA-Fehlerinformationen aufgezeichnet werden, wenn Diagnosestufe 3 oder höher in Kraft ist.

Sie sollten auch die Informationen der Konsolnachricht, der Fehlerdatei des Transaktionsmanagers oder andere produktspezifische Informationen über die verwendete externe Software zur Verarbeitung von Transaktionen berücksichtigen.

Der Datenbankmanager schreibt alle XA-spezifischen Fehler mit dem SLQCODE -998 (Fehler bei Transaktion oder heuristischer Maßnahme) und den entsprechenden Ursachencodes in das Serviceprotokoll des DB2-Diagnoseprogramms. Es folgen einige der häufigeren Fehlerursachen:

- Ungültige Syntax in der XA-Zeichenfolge zum Öffnen
- Fehler bei der Verbindung mit der in der XA-Zeichenfolge zum Öffnen angegebenen Datenbank aus einem der folgenden Gründe:
  - Die Datenbank ist nicht katalogisiert.
  - Die Datenbank wurde nicht gestartet.
  - Der Benutzername bzw. das Kennwort der Server-Anwendung ist zur Verbindung mit der Datenbank nicht berechtigt.
- Übertragungsfehler

Zugehörige Konzepte:

- „X/Open-Modell für die verteilte Transaktionsverarbeitung” auf Seite 205

Zugehörige Referenzen:

- „xa_open-Zeichenfolgeformate” auf Seite 213

Konfiguration von XA-Transaktionsmanagern

Konfigurieren von IBM WebSphere Application Server

IBM® WebSphere™ Application Server ist ein Java-basierter Anwendungsserver. Das Produkt kann die XA-Unterstützung von DB2 Universal Database™ (DB2 UDB) über die Java Transaction API (JTA) nutzen, die durch den JDBC-Treiber von DB2 bereitgestellt wird. Informationen zur Verwendung der Java Transaction API mit WebSphere Application Server sind der Dokumentation zu IBM WebSphere zu entnehmen. Die Dokumentation zu WebSphere Application Server kann online unter folgender Adresse eingesehen werden: http://www.ibm.com/software/webservers/appserv/infocenter.html.

Konfigurieren von IBM TXSeries CICS

Informationen zur Konfiguration von IBM TXSeries CICS® zur Verwendung von DB2 Universal Database™ (DB2 UDB) als Ressourcenmanager finden Sie im Handbuch *IBM TXSeries CICS Administration Guide*. Die TXSeries-Dokumentation steht online unter http://www.transarc.com/Library/documentation/websphere/WAS-EE/en_US/html/ zur Verfügung.

Host- und iSeries-Datenbankserver können an CICS-koordinierten Transaktionen teilnehmen.

Konfigurieren von IBM TXSeries Encina

Im Folgenden werden die verschiedenen APIs und Konfigurationsparameter aufgeführt, die für die Integration von Encina Monitor und DB2 Universal Database™-Servern (DB2 UDB) oder DB2 für z/OS und OS/390, DB2 für iSeries bzw. DB2 für VSE und VM erforderlich sind, wenn über DB2 Connect™ darauf zugegriffen wird. Die TXSeries-Dokumentation steht online unter http://www.transarc.com/Library/documentation/websphere/WAS-EE/en_US/html/ zur Verfügung.

Host- und iSeries-Datenbankserver können an Encina-koordinierten Transaktionen teilnehmen.

Konfigurieren von DB2 Universal Database

Gehen Sie wie folgt vor, um DB2 Universal Database™ (DB2 UDB) zu konfigurieren:

1. Jeder Datenbankname muss im DB2 UDB-Datenbankverzeichnis definiert werden. Wenn es sich bei der Datenbank um eine ferne Datenbank handelt, muss auch ein Eintrag im Knotenverzeichnis definiert werden. Sie können die Konfiguration über den Konfigurationsassistenten oder den DB2 UDB-Befehlszeilenprozessor ausführen. Zum Beispiel:

   ```
   DB2 CATALOG DATABASE inventdb AS inventdb AT NODE host1 AUTH SERVER
   DB2 CATALOG TCPIP NODE host1 REMOTE hostname1 SERVER svcname1
   ```

2. Der DB2 UDB-Client kann seine interne Verarbeitung für Encina optimieren, wenn er weiß, dass er es mit Encina zu tun hat. Sie können dies angeben, indem Sie den Konfigurationsparameter *tp_mon_name* des Datenbankmanagers auf `ENCINA` setzen. In der Standardeinstellung erfolgt keine spezielle Optimierung. Wenn *tp_mon_name* definiert ist, muss die Anwendung sicherstellen, dass der Thread, der die Arbeitseinheit ausführt, die Arbeit nach Beendigung auch sofort festschreibt. Es darf keine andere Arbeitseinheit gestartet werden. Wenn

dies *nicht* Ihre Umgebung ist, stellen Sie sicher, dass der Wert von *tp_mon_name* NONE ist (oder der über den Befehlszeilenprozessor auf NULL gesetzt wird). Der Parameter kann über die Steuerzentrale oder den Befehlszeilenprozessor aktualisiert werden. Der Befehl für den Befehlszeilenprozessor (CLP) lautet:

```
db2 update dbm cfg using tp_mon_name ENCINA
```

Konfigurieren von Encina für jeden Ressourcenmanager

Bei der Konfiguration von Encina für jeden Ressourcenmanager (RM) muss ein Administrator die Zeichenfolge zum Öffnen, die Zeichenfolge zum Schließen und den Thread der Steuerungsvereinbarung für jede DB2 UDB-Datenbank als Ressourcenmanager definieren, damit der Ressourcenmanager für Transaktionen in einer Anwendung registriert werden kann. Die Konfiguration kann mit der Enconcole-Gesamtanzeigeschnittstelle oder mit der Encina-Befehlszeilenschnittstelle ausgeführt werden. Zum Beispiel:

```
monadmin create rm inventdb -open "db=inventdb,uid=benutzer1,pwd=kennwort1"
```

Es gibt eine Ressourcenmanagerkonfiguration für jede DB2 UDB-Datenbank, und jede Konfiguration eines Ressourcenmanagers(RM) muss einen rm-Namen („logischen RM-Namen") haben. Dieser sollte zur Vereinfachung mit dem Datenbanknamen übereinstimmen.

Die XA-Zeichenfolge zum Öffnen (xa_open) enthält Informationen, die erforderlich sind, um eine Verbindung zur Datenbank herzustellen. Der Inhalt der Zeichenfolge ist RM-spezifisch. Die XA-Zeichenfolge zum Öffnen von DB2 UDB enthält den Aliasnamen der Datenbank, die geöffnet werden soll, und wahlweise eine Benutzer-ID und ein Kennwort, die der Verbindung zugeordnet werden sollen. Beachten Sie, dass der hier definierte Datenbankname auch in dem normalen Datenbankverzeichnis katalogisiert werden muss, das für alle Datenbankzugriffe erforderlich ist.

Die XA-Zeichenfolge zum Schließen (xa_close) wird von DB2 UDB nicht verwendet.

Der Thread der Steuerungsvereinbarung (Thread of Control Agreement) bestimmt, ob ein Anwendungsagent-Thread mehrere Transaktionen gleichzeitig verarbeiten kann.

Wenn Sie auf DB2 für z/OS und OS/390, DB2 für iSeries oder DB2 für VSE und VM zugreifen, müssen Sie den DB2-Synchronisationspunktmanager verwenden.

Verweisen auf eine DB2 UDB-Datenbank aus einer Encina-Anwendung

Gehen Sie wie folgt vor, um auf eine DB2 UDB-Datenbank aus einer Encina-Anwendung zu verweisen:

1. Verwenden Sie die Encina Scheduling Policy-API, um anzugeben, wie viele Anwendungsagenten aus einem einzelnen TP-Monitoranwendungsprozess ausgeführt werden können. Zum Beispiel:

   ```
   rc = mon_SetSchedulingPolicy (MON_EXCLUSIVE)
   ```

2. Verwenden Sie die Encina RM Registration-API, um den XA-Schalter und den logischen RM-Namen bereitzustellen, die von Encina zur Angabe des RMs in einem Anwendungsprozess verwendet werden sollen. Zum Beispiel:

   ```
   rc = mon_RegisterRmi ( &db2xa_switch,   /* xa-Schalter */
                          "inventdb",      /* logischer RM-Name */
                          &rmiId );        /* interne RM-ID */
   ```

Der XA-Schalter enthält die Adressen der XA-Routinen im RM, die der TM aufrufen kann. Er gibt auch die Funktionalität an, die vom RM bereitgestellt wird. Der XA-Schalter von DB2 UDB ist `db2xa_switch`, und er befindet sich in der DB2 UDB-Clientbibliothek (`db2app.dll` auf Windows-Betriebssystemen sowie in `libdb2` auf UNIX-basierten Systemen).

Von Encina wird der logische RM-Name und nicht der eigentliche Name der Datenbank verwendet, die von der SQL-Anwendung verwendet wird, die unter Encina ausgeführt wird. Der tatsächliche Name der Datenbank wird in der XA-Zeichenfolge zum Öffnen in der Encina RM Registration-API angegeben. Der logische RM-Name in diesem Beispiel stimmt mit dem Datenbanknamen überein.

Der dritte Parameter gibt eine interne Kennung zurück, die vom TM verwendet wird, um auf diese Verbindung zu verweisen.

Zugehörige Konzepte:

- „DB2 Connect und Transaktionsverarbeitungsmonitore" in *DB2 Connect Benutzerhandbuch*

Zugehörige Referenzen:

- „tp_mon_name - Name des Transaktionsprozessormonitors" in *Systemverwaltung: Optimierung*
- „xa_open-Zeichenfolgeformate" auf Seite 213

Konfigurieren von BEA Tuxedo

Vorgehensweise:

Führen Sie die folgenden Schritte aus, um Tuxedo zur Verwendung von DB2 Universal Database™ (DB2 UDB) als Ressourcenmanager zu konfigurieren:

1. Installieren Sie Tuxedo, wie in der Dokumentation für das Produkt angegeben. Stellen Sie sicher, dass Sie die gesamte Basiskonfiguration von Tuxedo durchführen, einschließlich Protokolldateien und Umgebungsvariablen.

   Sie benötigen auch einen Compiler und DB2 UDB Application Development Client. Installieren Sie diese, falls erforderlich.
2. Definieren Sie auf der Tuxedo-Server-ID die Umgebungsvariable DB2INSTANCE, so dass sie auf das Exemplar verweist, das die Datenbanken enthält, die Tuxedo verwenden soll. Definieren Sie die Variable PATH, so dass sie die Programmverzeichnisse von DB2 UDB enthält. Bestätigen Sie, dass die Tuxedo-Server-ID eine Verbindung zu den DB2 UDB-Datenbanken herstellen kann.
3. Aktualisieren Sie den Konfigurationsparameter *tp_mon_name* des Datenbankmanagers mit dem Wert `TUXEDO`.
4. Fügen Sie eine Definition für DB2 UDB zur Tuxedo-Ressourcenmanagerdefinitionsdatei hinzu. In den folgenden Beispielen ist `UDB_XA` der lokal definierte Tuxedo-Ressourcenmanagername für DB2 UDB und *db2xa_switch* der DB2-definierte Name für eine Struktur des Typs `xa_switch_t`:
   - Für AIX. Fügen Sie in der Datei `${TUXDIR}/udataobj/RM` folgende Definition hinzu:

     ```
     # DB2 UDB
     UDB_XA:db2xa_switch:-L${DB2DIR} /lib -ldb2
     ```

     Dabei ist {TUXDIR} das Verzeichnis, in dem Sie Tuxedo installiert haben, und {DB2DIR} das Verzeichnis des DB2 UDB-Exemplars.

- Für Windows NT. Fügen Sie in der Datei %TUXDIR%\udataobj\rm folgende Definition hinzu:

```
# DB2 UDB
UDB_XA;db2xa_switch;%DB2DIR%\lib\db2api.lib
```

  Dabei ist %TUXDIR% das Verzeichnis, in dem Sie Tuxedo installiert haben, und %DB2DIR% das Verzeichnis des DB2 UDB-Exemplars.

5. Erstellen Sie das Tuxedo-Transaktionsmonitor-Server-Programm für DB2 UDB:
   - Für AIX:

```
${TUXDIR}/bin/buildtms -r UDB_XA -o ${TUXDIR}/bin/TMS_UDB
```

     Dabei ist {TUXDIR} das Verzeichnis, in dem Sie Tuxedo installiert haben.
   - Für Windows NT:

```
%TUXDIR%\bin\buildtms -r UDB_XA -o %TUXDIR%\bin\TMS_UDB
```

6. Erstellen Sie die Anwendungsserver. In den folgenden Beispielen gibt die Option -r den Ressourcenmanagernamen, die Option -f (einmal oder mehrmals verwendet) die Dateien, die die Anwendungsservices enthalten, die Option -s die Anwendungsservicenamen für diesen Server und die Option -o den Ausgabe-Server-Dateinamen an:
   - Für AIX:

```
${TUXDIR}/bin/buildserver -r UDB_XA -f svcfile.o -s SVC1,SVC2
   -o UDBserver
```

     Dabei ist {TUXDIR} das Verzeichnis, in dem Sie Tuxedo installiert haben.
   - Für Windows NT:

```
%TUXDIR%\bin\buildserver -r UDB_XA -f svcfile.o -s SVC1,SVC2
   -o UDBserver
```

     Dabei ist %TUXDIR% das Verzeichnis, in dem Sie Tuxedo installiert haben.
7. Richten Sie die Tuxedo-Konfigurationsdatei so ein, dass auf den DB2 UDB-Server verwiesen wird. Fügen Sie im Abschnitt *GROUPS der Datei UDBCONFIG einen Eintrag wie den folgenden hinzu:

```
UDB_GRP   LMID=simp GRPNO=3
  TMSNAME=TMS_UDB TMSCOUNT=2
  OPENINFO="UDB_XA:db=sample,uid=db2_benutzer,pwd=db2_benutzer_kwt"
```

   Dabei gibt der Parameter TMSNAME das Transaktionsmonitor-Server-Programm an, das Sie zuvor erstellt haben, und der Parameter OPENINFO gibt den Ressourcenmanagernamen an. Darauf folgen der Datenbankname sowie die DB2 UDB-Benutzer-ID und das Kennwort, die für die Authentifizierung verwendet werden.

   Die Anwendungsserver, die Sie zuvor erstellt haben, werden im Abschnitt *SERVERS der Tuxedo-Konfigurationsdatei angegeben.
8. Wenn die Anwendung auf Daten zugreift, die sich in DB2 für z/OS und OS/390, DB2 für iSeries oder DB2 für VM&VSE befinden, ist der DB2 Connect XA-Konzentrator erforderlich.
9. Starten Sie Tuxedo:

```
tmboot -y
```

   Nach Beendigung des Befehls sollten Tuxedo-Nachrichten angeben, dass die Server gestartet wurden. Außerdem sollten Sie, wenn Sie den DB2 UDB-Befehl LIST APPLICATIONS ALL absetzen, zwei Verbindungen sehen, die (in diesem

Fall) vom Parameter TMSCOUNT in der UDB-Gruppe in der Tuxedo-Konfigurationsdatei `UDBCONFIG` angegeben werden.

Zugehörige Konzepte:

- „DB2 Connect und Transaktionsverarbeitungsmonitore" in *DB2 Connect Benutzerhandbuch*

Zugehörige Referenzen:

- „tp_mon_name - Name des Transaktionsprozessormonitors" in *Systemverwaltung: Optimierung*
- „LIST APPLICATIONS Command" in *Command Reference*

Teil 3. Anhänge und Schlussteil

Anhang A. Inkompatibilitäten zwischen Releases

Dieser Abschnitt beschreibt Inkompatibilitäten, die zwischen dem aktuellen Release von DB2 Universal Database™ (DB2 UDB) und früheren Releases von DB2 Universal Database bestehen.

Eine *Inkompatibilität* ist ein Bestandteil von DB2 Universal Database, dessen Funktionsweise gegenüber einem früheren Release von DB2 Universal Database geändert wurde. Bei Verwendung in einer vorhandenen Anwendung führt sie zu einem unerwarteten Ergebnis, macht eine Änderung in der Anwendung erforderlich oder beeinträchtigt die Leistung. In diesem Kontext bezieht sich die Bezeichnung *Anwendung* auf folgende Komponenten:

- Anwendungsprogrammcode
- Dienstprogramme anderer Hersteller
- Interaktive SQL-Abfragen
- Aufrufe von Befehlen oder APIs

Es werden Inkompatibilitäten beschrieben, die mit DB2 Universal Database Version 7 und Version 8 eingeführt wurden. Sie werden nach folgenden Kategorien untergliedert:

- Systemkataloginformationen
- Anwendungsprogrammierung
- SQL
- Datenbanksicherheit und Optimierung
- Dienstprogramme und Tools
- Konnektivität und Koexistenz
- Nachrichten
- Konfigurationsparameter

Jeder Inkompatibilitätsabschnitt einhält eine Beschreibung der Inkompatibilität, das Symptom oder den Effekt der Inkompatibilität und mögliche Lösungsmaßnahmen. Zu Anfang jeder Inkompatibilitätsbeschreibung gibt ein Indikator an, auf welches Betriebssystem sich die Inkompatibilität bezieht:

Windows
Von DB2 Universal Database unterstützte Microsoft Windows®-Plattformen

UNIX Von DB2 Universal Database unterstützte UNIX®-basierte Plattformen

OS/2 OS/2® (nur für DB2 Universal Database Version 7)

Geplante Inkompatibilitäten von DB2 Universal Database

Dieser Abschnitt beschreibt zukünftige Inkompatibilitäten, die Benutzer von DB2 Universal Database™ (DB2 UDB) bei der Codierung neuer Anwendungen bzw. der Änderung vorhandener Anwendungen berücksichtigen sollten. Dadurch wird die Migration auf künftige Versionen von DB2 UDB vereinfacht.

Systemkataloginformationen

PK_COLNAMES und FK_COLNAMES in einer künftigen Version von DB2 Universal Database

WIN	UNIX

Änderung: Die SYSCAT.REFERENCES-Spalten PK_COLNAMES und FK_COLNAMES sind nicht mehr verfügbar.

Symptom: Die Spalte ist nicht mehr vorhanden, und es wird ein Fehler zurückgegeben.

Erläuterung: Tools oder Anwendungen sind so codiert, dass sie die veralteten Spalten PK_COLNAMES und FK_COLNAMES verwenden.

Maßnahme: Ändern Sie das Tool oder die Anwendung so, dass stattdessen die Sicht SYSCAT.KEYCOLUSE verwendet wird.

COLNAMES in einer künftigen Version von DB2 Universal Database nicht mehr verfügbar

WIN	UNIX

Änderung: Die SYSCAT.INDEXES-Spalte COLNAMES ist nicht mehr verfügbar.

Symptom: Die Spalte ist nicht mehr vorhanden, und es wird ein Fehler zurückgegeben.

Erläuterung: Tools oder Anwendungen sind so codiert, dass sie die veraltete Spalte COLNAMES verwenden.

Maßnahme: Ändern Sie das Tool oder die Anwendung so, dass stattdessen die Sicht SYSCAT.INDEXCOLUSE verwendet wird.

Dienstprogramme und Tools

Unterstützung zur erneuten Erstellung von Indizes des Typs 1 wird entfernt

WIN	UNIX

Änderung: Mit Version 8 wird ein neuer Typ von Index unter der Bezeichnung Typ-2-Index (Type-2 Index) eingeführt. Bei Typ-1-Indizes, d. h. bei Indizes, die mit einer Version vor Version 8 erstellt wurden, wird ein Schlüssel im Rahmen der Löschung oder Aktualisierung einer Tabellenzeile physisch aus einer Blattseite entfernt. Bei Typ-2-Indizes werden Schlüssel als gelöscht markiert, wenn eine Zeile gelöscht oder aktualisiert wird, jedoch werden sie erst physisch entfernt, nachdem die Löschung oder die Aktualisierung festgeschrieben wurde. Wenn die Unterstützung für Typ-1-Indizes entfernt ist, müssen Sie Ihre Indizes nicht mehr manuell wiederherstellen. Typ-1-Indizes werden jedoch weiterhin ordnungsgemäß funktionieren. Durch alle Aktionen, die zur erneuten Erstellung von Indizes führen, werden Typ-1-Indizes automatisch in Typ-2-Indizes umgewandelt. In einer künftigen Version wird die Unterstützung für Typ-1-Indizes entfernt.

Erläuterung: Typ-2-Indizes besitzen Vorteile gegenüber Typ-1-Indizes:

- Ein Typ-2-Index kann über Spalten erstellt werden, die länger als 255 Byte sind.
- Die Nutzung von Sperren auf die nächsten Schlüssel ist auf ein Minimum beschränkt, wodurch sich der gemeinsame Zugriff verbessert.

Maßnahme: Entwickeln Sie einen Plan zur allmählichen Umwandlung Ihrer vorhandenen Indizes in Typ-2-Indizes. Die Funktion zur Onlineindexreorganisation kann Ihnen dabei helfen und gleichzeitig die Verfügbarkeitsausfälle minimieren. Vergrößern Sie den Tabellenbereich für Indizes, falls erforderlich. Ziehen Sie die Erstellung neuer Indizes in großen Tabellenbereichen und die Verlegung vorhandener Indizes in große Tabellenbereiche in Betracht.

Inkompatibilitäten von Version 8 zu vorigen Releases

Systemkataloginformationen

Spalte IMPLEMENTED in Katalogtabellen

Windows	UNIX

Änderung: In vorigen Versionen hatte die Spalte IMPLEMENTED in den Katalogtabellen SYSIBM.SYSFUNCTIONS und SYSCAT.SYSFUNCTIONS die Werte Y, M, H und N. In Version 8 sind die Werte Y und N.

Maßnahme: Codieren Sie Ihre Anwendungen um, so dass sie nur die Werte Y und N verwenden.

OBJCAT-Sichten in SYSCAT-Sichten umbenannt

Windows	UNIX

Änderung: Die folgenden OBJCAT-Sichten wurden in SYSCAT-Sichten umbenannt: TRANSFORMS, INDEXEXTENSIONS, INDEXEXTENSIONMETHODS, INDEXEXTENSIONDEP, INDEXEXTENSIONPARMS, PREDICATESPECS, INDEXEXPLOITRULES.

Maßnahme: Codieren Sie Ihre Anwendungen um, so dass sie die SYSCAT-Sichten verwenden.

SYSCAT-Sichten sind jetzt schreibgeschützt

Windows	UNIX

Änderung: Ab Version 8 lassen die SYSCAT-Sichten nur einen Lesezugriff zu.

Symptom: Eine UPDATE- oder INSERT-Operation an einer Sicht im Schema SYSCAT schlägt nun fehl.

Erläuterung: Die SYSSTAT-Sichten bilden den empfohlenen Weg zur Aktualisierung der Systemkataloginformationen. Einige SYSCAT-Sichten waren unbeabsichtigt als aktualisierbar implementiert. Dies wurde nun behoben.

Maßnahme: Ändern Sie Ihre Anwendungen, so dass sie anstelle der SYSCAT-Sichten auf aktualisierbare SYSSTAT-Sichten verweisen.

Anwendungsprogrammierung

Anweisungsgröße für Prüfkontextdatensätze wurde erhöht

Windows	UNIX

Änderung: Die Anweisungsbegrenzung wurde auf 2 MB angehoben.

Symptom: Der Anweisungstext für den Prüfkontextdatensatz ist zu groß, um in die Tabelle zu passen.

Erläuterung: Die vorhandenen Tabellen, die zur Aufzeichnung von Prüfkontextdatensätzen dienen, sehen nur 32 KB für den Anweisungstext vor. Die neue Begrenzung für Anweisungen beträgt 2 MB. Wenn Sie keine langen Anweisungen verwenden, betrifft Sie dies nicht.

Maßnahme: Erstellen Sie eine neue Tabelle zur Aufnahme der Prüfkontextdatensätze mit einem Wert CLOB(2M) für die Spalte des Anweisungstextes. Falls erwünscht, füllen Sie die neue Tabelle mit Daten aus der alten Tabelle, löschen dann Sie die alte Tabelle und verwenden die neue Tabelle. Die neue Tabelle kann in den gleichen Namen wie die alte Tabelle umbenannt werden. Binden Sie alle Anwendungen erneut, die die neue Tabelle verwenden.

Anwendungen werden standardmäßig im Multithreadmodus ausgeführt

Windows	UNIX

Änderung: In Version 8 werden Anwendungen standardmäßig im Multithreadmodus ausgeführt. In früheren Versionen wurden Anwendungen standardmäßig im Einzelthreadmodus ausgeführt. Diese Änderungen bedeutet, dass Aufrufe der API 'sqleSetTypeCtx' keine Wirkung haben.

Der Multithreadmodus von Version 8 ist äquivalent zum Aufruf der API 'sqleSetTypeCtx' mit der Option SQL_CTX_MULTI_MANUAL in einer Anwendung vor Version 8. Ein Client der Version 7 kann weiterhin eine Anwendung im Einzelthreadmodus ausführen.

Erläuterung: Wenn Sie in Version 7 eine Anwendung im Multithreadmodus ausführen wollten, mussten Sie Kontext-APIs aufrufen und die Kontexte verwalten. In Version 8 ist dies nicht erforderlich, da DB2 Universal Database™ (DB2 UDB) die Kontexte intern verwaltet. Wenn Sie dies wünschen, können Sie jedoch auch in Version 8 weiterhin Kontexte für Anwendungen mit Hilfe externer Kontext-APIs verwalten.

Fehler SQL0818N wird bei Verwendung der Option VERSION nicht zurückgegeben

Windows	UNIX

Änderung: Wenn Sie die neue Option VERSION in den Befehlen PRECOMPILE, BIND, REBIND und DROP PACKAGE verwenden, können Anforderung zur Ausführung nun einen Fehler SQL0805N anstelle des Fehlers SQL0818N zurückgeben.

Symptom: Anwendungen, deren Code eine Reaktion auf einen Fehler SQL0818N vorsieht, verhalten sich möglicherweise nicht mehr wie zuvor.

Maßnahme: Codieren Sie Ihre Anwendung so um, dass sie sowohl auf Fehler SQL0805N als auch auf Fehler SQL0818N reagieren kann.

Fehler SQL0306N wird nicht an Precompiler zurückgegeben, wenn eine Hostvariable nicht definiert ist

Windows	UNIX

Änderung: Wenn eine Hostvariable im Abschnitt BEGIN DECLARE nicht deklariert ist und im SQL-Abschnitt EXEC verwendet wird, wird vom Precompiler der Fehler SQL0306N nicht zurückgegeben. Wenn die Variable an anderer Stelle in der Anwendung deklariert ist, liefert die Anwendung bei der Ausführung SQL0804N zurück. Ist die Variable an keiner Stelle in der Anwendung deklariert, gibt der Compiler bei der Kompilierung einen Fehler zurück.

Symptom: Anwendungen, deren Code eine Reaktion auf einen Fehler SQL0306N bei der Vorkompilierung vorsieht, verhalten sich möglicherweise nicht mehr wie zuvor.

Maßnahme: Hostvariablen sollten im Abschnitt BEGIN DECLARE deklariert werden. Wenn Hostvariablen in einem anderen als dem Abschnitt BEGIN DECLARE deklariert werden, sollten Sie Ihre Anwendung so umcodieren, dass sie Rückkehrcodes SQL0804 verarbeiten kann.

Nicht unterstützte Datentypen für verschiebbare Cursor

Windows	UNIX

Änderung: Verschiebbare Cursors mit den Datentypen LONG VARCHAR, LONG VARGRAPHIC, DATALINK und LOB, einzigartigen Typen für diese Typen oder strukturierten Typen werden in Version 8 nicht unterstützt. Jeder dieser Datentypen, die für verschiebbare Cursor der Version 7 unterstützt werden, wird nicht länger unterstützt.

Symptom: Wenn Spalten mit diesen Datentypen in der SELECT-Liste eines verschiebbaren Cursors angegeben werden, wird Nachricht SQL0270N mit Ursachencode 53 zurückgegeben.

Maßnahme: Modifizieren Sie die SELECT-Liste des verschiebbaren Cursors, so dass sie keine Spalte mit einem dieser Datentypen enthält.

Euro-Version der Codepagekonvertierungstabellen

Windows	UNIX

Änderung: Die Codepagekonvertierungstabellen der Version 8, die Unterstützung für das Euro-Symbol bereitstellen, unterscheiden sich geringfügig von den Konvertierungstabellen, die mit früheren Versionen von DB2 UDB geliefert wurden.

Maßnahme: Für den Fall, dass Sie die Codepagekonvertierungstabellen einer vorigen Version (vor Version 8) verwenden wollen, werden diese im Verzeichnis `sqllib/conv/v7` bereitgestellt.

Umschalten zwischen LOB-Querverweis und einem LOB-Wert

Windows	UNIX

Änderung: Die Möglichkeit, zwischen einem LOB-Querverweis (LOB - großes Objekt) und einem LOB-Wert umzuschalten, wurde während des Bindens für eine Cursoranweisung geändert. Wenn eine Anwendung mit SQLRULES DB2 (Standardverhalten) gebunden wird, ist der Benutzer nicht in der Lage, zwischen LOB-Querverweisen und LOB-Werten umzuschalten.

Maßnahme: Wenn Sie zwischen einem LOB-Querverweis und einem LOB-Wert während des externen Bindens (Bindout) einer Cursoranweisung umschalten wollen, kompilieren Sie Ihre Anwendung mit der Angabe SQLRULES STD vor.

Nicht festgeschriebene Arbeitseinheiten auf UNIX-Plattformen

	UNIX

Änderung: In Version 8 führen alle Anwendungsbeendigungen zu einem impliziten Rückgängigmachen (ROLLBACK) für die ausstehenden Arbeitseinheiten. Windows-basierte Anwendungen ändern sich dadurch nicht, da sie bereits eine implizite ROLLBACK-Operation bei normalen oder abnormalen Beendigungen von Anwendungen durchführen. Vor Version 8 schrieben UNIX-basierte Anwendungen, die nicht entweder eine explizite oder eine implizite Kontextunterstützung verwendeten, eine ausstehende Arbeitseinheit fest (COMMIT), wenn die Anwendung normal beendet wurde, ohne direkt eine Anweisung CONNECT RESET, COMMIT oder ROLLBACK aufzurufen. CLI-, ODBC- und Java-basierte Anwendungen (implizite Kontextunterstützung) und Anwendungen, die explizit Anwendungskontexte erstellen, machten eine ausstehende Arbeitseinheit immer rückgängig, wenn die Anwendung beendet wurde. Eine abnormale Beendigung einer Anwendung führte ebenfalls zu einem impliziten Rückgängigmachen (ROLLBACK) für die ausstehende Arbeitseinheit.

Maßnahme: Um sicherzustellen, dass Transaktionen festgeschrieben werden (COMMIT), sollte die Anwendung entweder eine explizite COMMIT-Operation oder eine CONNECT RESET-Operation durchführen, bevor sie beendet wird.

Änderung an der Sicherungspunktbenennung

Windows	UNIX

Änderung: Namen von Sicherungspunkten dürfen nicht mehr mit ″SYS″ beginnen.

Symptom: Die Erstellung eines Sicherungspunkts, dessen Name mit ″SYS″ beginnt, schlägt mit Fehler SQL0707N fehl.

Erläuterung: Sicherungspunktnamen, die mit ″SYS″ beginnen, sind für die Verwendung durch das System reserviert.

Maßnahme: Benennen Sie alle Sicherungspunkte, die mit ″SYS″ beginnen, in einen anderen Namen um, der nicht mit ″SYS″ beginnt.

Fehler bei Codepagekonvertierungen und Bytesubstitutionen

Windows	UNIX

Änderung: Zeichendaten in Host-Eingabevariablen werden nötigenfalls in die Datenbankcodepage konvertiert bevor sie in der SQL-Anweisung verwendet werden, in der die Hostvariable erscheint. Während der Codepagekonvertierung kann es zu einer Datenerweiterung kommen. Früher wurde, wenn für Daten in einer Hostvariable eine Codepagekonvertierung festgestellt wurde, die tatsächliche angenommene Länge für die Hostvariable vergrößert, um die Erweiterung handzuhaben. Diese angenommene Vergrößerung der Länge wird nicht mehr vorgenommen, um die Auswirkungen der Änderung der Datentyplänge bei anderen SQL-Operationen zu verringern.

Anmerkung: Für Hostvariablen, die im Zusammenhang mit FOR BIT DATA verwendet werden, gilt das oben Genannte nicht. Die Daten in diesen Hostvariablen werden nicht konvertiert bevor sie für Bitdaten verwendet werden.

Symptom: Wenn die Hostvariable nicht groß genug ist, um die erweiterte Länge nach der Codepagekonvertierung zu enthalten, wird ein Fehler (SQLSTATE 22001, SQLCODE -302) zurückgegeben.

Erläuterung: Da es bei der Codepagekonvertierung zu einer Erweiterung bzw. einer Kontraktion der Daten kommen kann, können Operationen, die von der Länge der Daten in der Hostvariable abhängig sind, andere Ergebnisse liefern oder zu einer Fehlersituation führen.

Maßnahme: Die folgenden Alternativen können in Betracht gezogen werden:

- Die Anwendung kann codiert werden, um die Möglichkeit der Codepagekonvertierung handzuhaben, die die Änderung der Länge der Daten verursacht, indem die Länge der Hostzeichenvariablen vergrößert wird.
- Die Daten können geändert werden, um Zeichen zu umgehen, die die Erweiterung verursachen
- Die Anwendungscodepage kann so geändert werden, dass sie mit der Datenbankcodepage übereinstimmt. Auf diese Weise tritt keine Codepagekonvertierung auf.

Codepagekonvertierung für Hostvariablen

Windows	UNIX

Änderung: Eine erforderliche Codepagekonvertierung wird nun während der Einbindephase (Bind in) durchgeführt.

Symptom: Andere Ergebnisse.

Erläuterung: Da nun die Codepagekonvertierung für Hostvariablen bei Bedarf immer durchgeführt wird, findet die Auswertung von Vergleichselementen immer in der Datenbankcodepage und nicht in der Anwendungscodepage statt. Beispiel:

```
SELECT * FROM tabelle WHERE :hv1 > :hv2
```

Die Auswertung dieser Anweisung wird in der Datenbankcodepage und nicht in der Anwendungscodepage durchgeführt. Die verwendete Sortierfolge ist weiterhin die Sortierfolge der Datenbank.

Maßnahme: Prüfen Sie, ob die Ergebnisse in früheren Versionen tatsächlichen die gewünschten Ergebnisse waren. Ist dies der Fall, ändern Sie das Vergleichselement so, dass bei Verwendung der Sortierfolge und der Codepage der Datenbank die gewünschten Ergebnisse erzielt werden. Alternativ können Sie die Anwendungscodepage oder die Datenbankcodepage ändern.

Erweiterung und Kontraktion von Daten in Hostvariablen

Windows	UNIX

Änderung: Eine erforderliche Codepagekonvertierung wird nun während einer Bindeoperation durchgeführt.

Symptom: Daten aus Hostvariablen haben eine andere Länge.

Erläuterung: Da es bei der Codepagekonvertierung zu einer Erweiterung bzw. einer Kontraktion der Daten kommen kann, können Operationen, die von der Länge der Daten in der Hostvariable abhängig sind, andere Ergebnisse liefern oder zu einer Fehlersituation führen.

Maßnahme: Ändern Sie die Daten, die Anwendungscodepage oder die Datenbankcodepage, so dass die Codepagekonvertierung keine Änderungen der Länge der konvertierten Daten zur Folge hat, oder codieren Sie die Anwendung so, dass sie der Möglichkeit Rechnung trägt, dass sich die Länge der Daten aufgrund einer Codepagekonvertierung ändert.

Länge von Hostvariablen nach einer Codepagekonvertierung

Windows	UNIX

Änderung: Eine Codepagekonvertierung verursacht keine Verlängerung des Ergebnisses für Hostvariablen oder Parametermarken wegen Erweiterung mehr.

Symptom: Fehler durch Abschneiden von Daten.

Erläuterung: Die für die untypisierte Parametermarke festgelegte Länge des Zeichendatentyps wird nicht mehr verlängert, um eine potenzielle Erweiterung durch Codepagekonvertierung zu berücksichtigen. Die Ergebnislänge ist nun bei Operationen kürzer, die die Ergebnislänge anhand der Länge der untypisierten Parametermarke bestimmen. Zum Beispiel sei C1 eine Spalte des Typs CHAR(10):

```
VALUES CONCAT (?, C1)
```

Dieser Ausdruck hat nicht mehr den Ergebnisdatentyp und die Länge CHAR(40) für eine Datenbank, in der bei einer Konvertierung von der Anwendungscodepage in die Datenbankcodepage eine dreifache Erweiterung möglich ist, sondern erhält den Ergebnisdatentyp und die Länge CHAR(20).

Maßnahme: Verwenden Sie eine CAST-Spezifikation, um der untypisierten Parametermarke den gewünschten Typ zu geben, oder ändern Sie den Operanden, der den Typ der untypisierten Parametermarke bestimmt, in einen Datentyp bzw. eine Länge, die eine mögliche Erweiterung der Daten durch eine Codepagekonvertierung auffangen könnte.

Änderung an der Ausgabe der Anweisung DESCRIBE

Windows	UNIX

Änderung: Eine Codepagekonvertierung verursacht keine Verlängerung des Ergebnisses für Hostvariablen oder Parametermarken wegen Erweiterung mehr.

Symptom: Die Ausgabe der Anweisung DESCRIBE ändert sich.

Erläuterung: Da die Ergebnislänge durch eine potenzielle Erweiterung bei einer Codepagekonvertierung nicht verlängert wird, ist die Ausgabe einer Anweisung DESCRIBE, die eine solche Ergebnislänge beschreibt, nun unterschiedlich.

Maßnahme: Ändern Sie bei Bedarf die Anwendung so, dass sie die neuen Werte verarbeiten kann, die von der Anweisung DESCRIBE zurückgegeben werden.

Fehler bei der Verwendung der Funktion SUBSTR mit Hostvariablen

Windows	UNIX

Änderung: Eine Codepagekonvertierung verursacht keine Verlängerung des Ergebnisses für Hostvariablen oder Parametermarken wegen Erweiterung mehr.

Symptom: Fehler SQL0138N von der Funktion SUBSTR.

Erläuterung: Eine potenzielle Erweiterung durch eine Codepagekonvertierung wurde dadurch aufgefangen, dass die Länge, die für die Hostvariable vorgesehen wurde, vergrößert wurde. Dadurch konnte zum Beispiel ein Funktionsausdruck wie `SUBSTR (:hv,19,1)` erfolgreich für eine Hostvariable mit einer Länge von 10 ausgeführt werden. Dies funktioniert jetzt nicht mehr.

Maßnahme: Geben Sie eine größere Länge für die Hostvariable an, um die Länge der konvertierten Daten zu berücksichtigen, oder ändern Sie den SUBSTR-Aufruf so, dass Positionen innerhalb der Länge der Hostvariablen angegeben werden.

Nicht threadsichere Bibliotheken werden unter Solaris nicht mehr unterstützt

	UNIX

Änderung: Die nicht threadsichere (engl. non-thread safe) Bibliothek `libdb2_noth.so` ist nicht länger verfügbar.

Symptom: Tools oder Anwendungen, für die `libdb2_noth.so` erforderlich ist, funktionieren nicht mehr.

Erläuterung: Da die Unterstützung für die veralteten nicht threadsicheren Bibliotheken nicht mehr erforderlich ist, gehört die Bibliothek `libdb2_noth.so` nicht mehr zu DB2 UDB für Solaris Operating Environment™.

Maßnahme: Ändern Sie das Tool oder die Anwendung so, dass stattdessen die threadsichere Bibliothek `libdb2.so` verwendet wird. Linken Sie Ihre Anwendungen mit dem Parameter -mt erneut.

Importieren oder Exportieren von DBCLOB-Daten bei Verbindung zu einer Unicode-Datenbank

Windows	UNIX

Änderung: Wenn Sie vor Version 8 Daten exportierten, die DBCLOB-Daten aus einer Unicode-Datenbank (UTF-8) enthielten, und die Angabe LOBSINFILE zur Dateitypänderung verwendeten, wurden die DBCLOB-Daten in Codepage 1200 (Unicode-Grafikcodepage) exportiert. Wenn Sie Daten importierten, die DBCLOB-Daten enthielten, und die Dateitypänderung durch LOBSINFILE verwendeten, wurden die DBCLOB-Daten in Codepage 1200 (Unicode-Grafikcodepage) importiert. Diese Funktionsweise wird in Version 8 beibehalten, wenn Sie die Registriervariable DB2GRAPHICUNICODESERVER auf den Wert ON setzen.

In Version 8 hat die Standardeinstellung der Registriervariablen DB2GRAPHICUNICODESERVER den Wert OFF. Wenn Sie Daten, die DBCLOB-Daten enthalten, exportieren und die Datentypänderung durch LOBSINFILE verwenden, werden die DBCLOB-Daten in der Grafikcodepage der Anwendung exportiert. Wenn Sie Daten, die DBCLOB-Daten enthalten, importieren und die Datentypänderung durch LOBSINFILE verwenden, werden die DBCLOB-Daten in der Grafikcodepage der Anwendung importiert. Wenn die Codepage der Anwendung IBM-eucJP (954) oder IBM-eucTW (964) ist und Sie Daten mit DBCLOB-Daten unter Verwendung der Datentypänderung durch LOBSINFILE exportieren, werden die DBCLOB-Daten in der Zeichencodepage der Anwendung exportiert. Wenn Sie Daten mit DBCLOB-Daten importieren und die Datentypänderung durch LOBSINFILE verwenden, werden die DBCLOB-Daten in der Zeichencodepage der Anwendung importiert.

Symptom: Beim Importieren von Daten mit der Datentypänderung durch LOBSINFILE in eine Unicode-Datenbank werden die Zeichendaten korrekt konvertiert, die DBCLOB-Daten werden jedoch beschädigt.

Maßnahme: Wenn Sie Daten zwischen einer Datenbank der Version 8 und einer früheren Datenbank versetzen, setzen Sie die Registriervariable DB2GRAPHICUNICODESERVER auf den Wert ON, um die frühere Arbeitsweise beizubehalten.

SQL

Identische spezifische Namen für Funktionen und Prozeduren unzulässig

Windows	UNIX

Änderung: Der Namensbereich für SPECIFICNAME wurde vereinigt. In früheren Versionen von DB2 UDB war es zulässig, dass eine Funktion und eine Prozedur denselben spezifischen Namen hatten. In Version 8 ist dies jedoch nicht zulässig.

Symptom: Bei der Migration einer Datenbank auf Version 8 prüft das Dienstprogramm db2ckmig auf Funktionen und Prozeduren mit dem gleichen spezifischen Namen. Wenn bei der Migration doppelte Namen festgestellt werden, schlägt die Migration fehl.

Maßnahme: Löschen Sie die Prozedur und erstellen Sie sie mit einem anderen spezifischen Namen neu.

Zugriffsrecht EXECUTE für Funktionen und Prozeduren

Windows	UNIX

Änderung: Früher musste ein Benutzer eine Routine nur erstellen, damit andere sie verwenden konnten. Nun muss ein Benutzer nach der Erstellung einer Routine zunächst den Befehl GRANT EXECUTE für sie ausführen, damit andere die Routine verwenden können.

In früheren Versionen gab es keine Berechtigungsprüfungen für Prozeduren, jedoch musste der aufrufende Benutzer das Zugriffsrecht EXECUTE für jedes Paket besitzen, das aus der Prozedur heraus aufgerufen wurde. Für eine eingebettete Anwendung, die mit der Angabe CALL_RESOLUTION IMMEDIATE in Version 8, und für eine katalogisierte CLI-Prozedur muss der aufrufende Benutzer das Zugriffsrecht EXECUTE für die Prozedur haben. Nur der definierende Benutzer der Prozedur muss das Zugriffsrecht EXECUTE für alle Pakete haben.

Symptom:

1. Eine Anwendung funktioniert möglicherweise nicht korrekt.
2. Eine vorhandene Prozedur, die aus mehreren Paketen besteht und für die der definierende Benutzer keinen Zugriff auf alle Pakete hat, funktioniert nicht korrekt.

Maßnahme:

1. Setzen Sie die erforderlichen Anweisungen GRANT EXECUTE ab. Wenn sich alle Routinen in einem einzigen Schema befinden, können die Zugriffsrechte für jeden Typ von Routine mit einer einzigen Anweisung erteilt werden. Zum Beispiel:

   ```
   GRANT EXECUTE ON FUNCTION schema1.* TO PUBLIC
   ```

2. Wenn ein Paket durch jeden verwendbar ist, jedoch ein anderes Paket auf wenige berechtigte Benutzer beschränkt ist, reagiert eine gespeicherte Prozedur, die mit beiden Paketen arbeitet, auf einen Berechtigungsfehler, wenn sie versucht, auf das zweite Paket zuzugreifen. Wenn sie den Berechtigungsfehler erkennt, ist ihr bekannt, dass der Benutzer kein berechtigter Benutzer ist, und die Prozedur überspringt einen Teil ihrer Logik.

   Sie können dem auf verschiedene Weise Rechnung tragen:

   a. Beim Vorkompilieren eines Programm sollte CALL_RESOLUTION DEFERRED definiert werden, um anzuzeigen, dass das Programm als Aufruf der veralteten API sqleproc() ausgeführt wird, wenn der Precompiler eine Prozedur in einer Anweisung CALL nicht auflösen kann.
   b. Das CLI-Schlüsselwort UseOldStpCall kann der Datei db2cli.ini hinzugefügt werden, um die Methode anzugeben, mit der Prozeduren aufgerufen werden. Es kann zwei Werte haben: Der Wert *0* bedeutet, dass Prozeduren nicht mit der alten Aufrufmethode aufgerufen werden, während der *1* bedeutet, dass Prozeduren mit der alten Aufrufmethode aufgerufen werden.
   c. Erteilen Sie das Zugriffsrecht EXECUTE jedem, der das Paket ausführt.

Hinzufügen einer Fremdschlüsselbedingung zu einer Tabelle

Windows	UNIX

Änderung: Wenn Sie in früheren Versionen einen Fremdschlüssel definiert haben, der auf eine Tabelle im Status 'Überprüfung anstehend' verwies, wurde die abhän-

gige Tabelle ebenfalls in den Status 'Überprüfung anstehend' versetzt. Wenn Sie in Version 8 einen Fremdschlüssel erstellen, der auf eine Tabelle im Status 'Überprüfung anstehend' verweist, sind zwei Ergebnisse möglich:

1. Wenn der Fremdschlüssel bei der Erstellung der abhängigen Tabelle hinzugefügt wird, sind das Erstellen der Tabelle und das Hinzufügen der Integritätsbedingung erfolgreich, weil die Tabelle leer erstellt wird und somit keine Zeilen die Integritätsbedingung verletzen.
2. Wenn einer vorhandenen Tabelle ein Fremdschlüssel hinzugefügt wird, empfangen Sie Fehler SQL0668N.

Maßnahme: Aktivieren Sie mit Hilfe der Anweisung SET INTEGRITY ... IMMEDIATE CHECKED die Integritätsprüfung für die Tabelle, die sich im Status 'Überprüfung anstehend' befindet', bevor Sie den Fremdschlüssel hinzufügen, der auf die Tabelle verweist.

Änderung an SET INTEGRITY ... IMMEDIATE CHECKED

Windows	UNIX

Änderung: In früheren Releases wurde eine Tabelle, für die die Anweisung SET INTEGRITY ... UNCHECKED abgesetzt wurde (d. h. mit einigen Byte 'U' in der Spalte const_checked der Katalogtabelle SYSCAT.TABLES), standardmäßig bei der nächsten Anweisung SET INTEGRITY ... IMMEDIATE CHECKED vollständig verarbeitet, was bedeutete, dass sämtliche Datensätze auf Verletzungen von Integritätsbedingungen überprüft wurden. Sie mussten explizit das Schlüsselwort INCREMENTAL angeben, um eine vollständige Verarbeitung zu vermeiden.

Wenn in Version 8 die Anweisung SET INTEGRITY ... IMMEDIATE CHECKED abgesetzt wird, besteht die Standardfunktionsweise darin, die ungeprüften Daten zu belassen (d. h. die Byte 'U' beizubehalten), indem nur eine inkrementelle Verarbeitung ausgeführt wurde. (Eine Warnung wird zurückgegeben, dass alte Daten ungeprüft bleiben.)

Erläuterung: Diese Änderung wurde vorgenommen, um zu vermeiden, dass die Standardfunktionsweise in einer Überprüfung der Integritätsbedingungen für alle Datensätze besteht, was in der Regel mehr Ressourcen beansprucht.

Maßnahme: Sie müssen explizit NOT INCREMENTAL angeben, um eine vollständige Verarbeitung zu erzwingen.

Dezimaltrennzeichen für die Funktion CHAR

Windows	UNIX

Änderung: Dynamische Anwendungen, die auf Servern mit einer Ländereinstellung (Locale) ausgeführt werden, in der das Komma als Dezimaltrennzeichen definiert ist, und die unqualifizierte Aufrufe der Funktion CHAR mit einem Argument des Typs REAL oder DOUBLE enthalten, geben einen Punkt als Trennzeichen im Ergebnis der Funktion CHAR(double) zurück. Diese Inkompatibilität wird auch sichtbar, wenn Objekte wie Sichten und Auslöser in Version 8 erneut erstellt werden oder wenn statische Pakete explizit erneut gebunden werden.

Erläuterung: Dies ist ein Ergebnis der Auflösung der neuen Funktionskennung SYSIBM.CHAR(double) anstelle der Signatur SYSFUN.CHAR(double).

Maßnahme: Zur Beibehaltung der Funktionsweise aus früheren Versionen von DB2 UDB muss die Anwendung die Funktion explizit mit SYSFUN.CHAR aufrufen und darf der Funktionsauflösung nicht die Auswahl der Signatur SYSIBM-.CHAR ermöglichen.

Änderungen an der Anweisung CALL

Windows	UNIX

Änderung: In Version 8 haben eine Anwendung, die mit CALL_RESOLUTION IMMEDIATE vorkompiliert wird, und eine katalogisierte CLI-Prozedur mehrere wichtige Unterschiede im Vergleich zu früheren Versionen:

- Die Unterstützung für Hostvariablen wurde durch eine Unterstützung für dynamische CALL-Aufrufe ersetzt.
- Die Unterstützung für die Kompilierung von Anwendungen, die nicht katalogisierte gespeicherte Prozeduren aufrufen, wurde entfernt. Die Unterstützung für nicht katalogisierte gespeicherte Prozeduren wird in einem zukünftigen Release von DB2 UDB vollständig entfernt.
- Die Unterstützung für gespeicherte Prozeduren mit variabler Argumentliste wurde für veraltet erklärt.
- Es gelten andere Regeln für das Laden der Bibliothek der gespeicherten Anwendungen.

Maßnahme: Die Anweisung CALL, wie sie vor Version 8 unterstützt wurde, wird weiterhin verfügbar sein. Auf sie kann mit Hilfe der Option CALL_RESOLUTION DEFERRED im Befehl PRECOMPILE PROGRAM zugegriffen werden.

Vorhandene Anwendungen (die vor Version 8 erstellt wurden) werden weiterhin funktionieren. Wenn Anwendungen ohne die Option CALL_RESOLUTION DEFERRED erneut vorkompiliert werden, sind möglicherweise Quellcodeänderungen erforderlich.

Die Unterstützung für die Anweisung CALL_RESOLUTION DEFERRED wird in einer zukünftigen Version entfernt.

Ausgabe aus UDFs liefert Zeichenfolgen fester Länge

Windows	UNIX

Änderung: Eine benutzerdefinierte Funktion (UDF, Skalar- oder Tabellenfunktion) kann so definiert werden, dass sie eine Zeichenfolge (CHAR(n) oder GRAPHIC(n)) mit fester Länge zurückgibt. Wenn in früheren Versionen der zurückgegebene Wert ein eingebettetes Nullzeichen enthielt, war das Ergebnis einfach n Byte (bzw. 2n Byte für GRAPHIC-Datentypen), einschließlich des Nullzeichens und alle Byte rechts neben dem Nullzeichen. In Version 8 sucht DB2 UDB nach dem Nullzeichen und gibt Leerzeichen von der Stelle (des Nullzeichens) an bis zum Ende des Werts zurück.

Maßnahme: Wenn Sie weiterhin die Funktionsweise der Versionen vor Version 8 wünschen, ändern Sie die Definition des zurückgegebenen Werts von CHAR(n) in CHAR(n) FOR BIT DATA. Für GRAPHIC-Daten steht keine Methode zur weiteren Verwendung der Funktionsweise vor Version 8 zur Verfügung.

Änderung in der Funktionsweise der Datenbankverbindung

Windows	UNIX

Änderung: Wenn Sie in Version 7 durch eingebettetes SQL eine Verbindung zu einer Datenbank herstellen, und dann versuchen, eine Verbindung zu einer nicht vorhandenen Datenbank herzustellen, schlägt der Versuch, die Verbindung zu der nicht vorhandenen Datenbank herzustellen, mit SQL1013N fehl. Die Verbindung zur ersten Datenbank bleibt bestehen. In Version 8 führt der Versuch, eine Verbindung zu der nicht vorhandenen Datenbank herzustellen, zu einer Trennung der Verbindung zur ersten Datenbank. Dies hat zur Folge, dass die Anwendung anschließend über keine Verbindung verfügt.

Maßnahme: Codieren Sie das eingebettete SQL so, dass die Verbindung zur ersten Datenbank im Anschluss an einen fehlgeschlagenen Versuch, eine Verbindung zu einer anderen Datenbank herzustellen, erneut hergestellt wird.

Widerrufen des Zugriffsrechts CONTROL für Pakete

Windows	UNIX

Änderung: Mit Hilfe des Zugriffsrechts CONTROL kann ein Benutzer Zugriffsrechte für ein Paket erteilen. In DB2 UDB Version 8 stellt die Angabe WITH GRANT OPTION einen Mechanismus dar, die Berechtigung eines Benutzers zur Erteilung von Zugriffsrechten für Pakete an andere Benutzer festzulegen. Dieser Mechanismus wird anstelle des Zugriffsrechts CONTROL verwendet, um festzulegen, ob ein Benutzer anderen Zugriffsrechte erteilen darf. Wenn das Zugriffsrecht CONTROL widerrufen wird, können Benutzer weiterhin anderen Benutzern Zugriffsrechte erteilen.

Symptom: Ein Benutzer kann auch nach dem Widerruf des Zugriffsrechts CONTROL weiterhin Zugriffsrechte für ein Paket erteilen.

Maßnahme: Wenn ein Benutzer nicht berechtigt sein soll, Zugriffsrechte für Pakete anderen Benutzern zu erteilen, widerrufen Sie alle Zugriffsrechte für das Paket und erteilen nur die erforderlichen Zugriffsrechte.

Fehler beim Umsetzen (CAST) einer FOR BIT DATA-Zeichenfolge in einen CLOB-Datentyp

Windows	UNIX

Änderung: Die Umsetzung einer Zeichenfolge, die mit FOR BIT DATA definiert ist, in einen CLOB-Datentyp (mit der CAST-Spezifikation oder der Funktion CLOB) gibt nun einen Fehler (SQLSTATE 42846) zurück.

Symptom: Die Umsetzung in einen CLOB-Datentyp gibt nun einen Fehler in Fällen zurück, in denen dies zuvor nicht geschah.

Erläuterung: FOR BIT DATA wird für den Datentyp CLOB nicht unterstützt. Das Ergebnis der Verwendung der CAST-Spezifikation oder der Funktion CLOB, wenn eine FOR BIT DATA-Zeichenfolge als Argument übergeben wird, ist nicht definiert. Dieser Fall wird nun als Fehler abgefangen.

Maßnahme: Ändern Sie das Argument für die CAST-Spezifikation oder die Funktion CLOB, so dass es keine FOR BIT DATA-Zeichenfolge ist. Dies können Sie erreichen, indem Sie die FOR BIT DATA-Zeichenfolge mit Hilfe der CAST-Spezifikation in eine FOR SBCS DATA-Zeichenfolge oder eine FOR MIXED DATA-Zeichenfolge umsetzen. Wenn zum Beispiel C1FBD eine Spalte des Typs VARCHAR(20) ist, die mit FOR BIT DATA deklariert ist, stellt der folgende Ausdruck in einer Nicht-DBCS-Datenbank ein gültiges Argument für die Funktion CLOB dar:

```
CAST (C1FBD AS VARCHAR(20) FOR SBCS DATA)
```

Ausgabe der Funktion CHR

Windows	UNIX

Änderung: Die Funktion CHR(0) liefert ein Leerzeichen (X'20') anstelle des Zeichens mit Codepunkt X'00'.

Symptom: Die Ausgabe der Funktion CHR mit dem Wert X'00' als Argument liefert andere Ergebnisse.

Erläuterung: Die Zeichenfolgebearbeitung beim Aufrufen und Zurückkehren von benutzerdefinierten Funktionen interpretiert den Wert X'00' als Ende einer Zeichenfolge.

Maßnahme: Ändern Sie den Anwendungscode, um den neuen Ausgabewert entsprechend zu verarbeiten. Alternativ können Sie auch eine benutzerdefinierte Funktion definieren, die als Ergebnis CHAR(1) FOR BIT DATA zurückgibt und auf der SYSFUN-Funktion CHR als Quellenfunktion beruht, und diese Funktion vor SYSFUN im SQL-Pfad angeben.

Funktionen TABLE_NAME und TABLE_SCHEMA können nicht in generierten Spalten oder Prüfungen auf Integritätsbedingung verwendet werden

Windows	UNIX

Änderung: Die Definitionen für die Funktionen TABLE_NAME und TABLE_SCHEMA wurden korrigiert und können jetzt nicht mehr in generierten Spalten oder Prüfungen auf Integritätsbedingung verwendet werden.

Symptom: Das Binden schlägt mit einem SQLCODE-Wert -548/SQLSTATE-Wert 42621 fehl, der angibt, dass TABLE_NAME oder TABLE_SCHEMA im Kontext einer Prüfung auf Integritätsbedingung ungültig ist.

Erläuterung: Die Funktionen TABLE_NAME und TABLE_SCHEMA rufen Daten aus Katalogsichten ab. Sie gehören zur Klasse READS SQL DATA. Funktionen der Klasse READS SQL DATA sind in Ausdrücken GENERATED COLUMN und in Prüfungen auf Integritätsbedingung nicht zulässig, weil DB2 UDB keine zu jedem Zeitpunkt beständige Korrektheit der Integritätsbedingung erzwingen kann.

Maßnahme: Aktualisieren Sie alle Spalten, die generierte Spaltenausdrücke und Prüfungen auf Integritätsbedingung enthalten, um die Verwendung von TABLE_NAME und TABLE_SCHEMA zu entfernen. Verwenden Sie die Anweisung ALTER TABLE zum Festlegen (SET) eines neuen Ausdrucks, um eine generierte Spalte zu ändern. Verwenden Sie die Anweisung ALTER TABLE mit der Klausel

DROP CONSTRAINT, um eine Prüfung auf Integritätsbedingung zu entfernen. Dadurch können Sie die Tabellen binden, die die betreffenden Spalten enthalten, und weiter darauf zugreifen.

Datenbanksicherheit und Optimierung

Berechtigung für die Anweisungen CREATE FUNCTION, CREATE METHOD und CREATE PROCEDURE

Windows	UNIX

Änderung: Mit Version 8 wird die Berechtigung CREATE_EXTERNAL_ROUTINE eingeführt.

Symptom: Die Anweisungen CREATE FUNCTION, CREATE METHOD und CREATE PROCEDURE mit der Option EXTERNAL schlagen möglicherweise fehl.

Maßnahme: Erteilen Sie Benutzern, die die Anweisungen CREATE FUNCTION, CREATE METHOD und CREATE PROCEDURE mit der Option EXTERNAL absetzen, die Berechtigung CREATE_EXTERNAL_ROUTINE.

Dienstprogramme und Tools

Änderungen beim Überwachen der Leistung über die Steuerzentrale

Windows	UNIX

Symptom: Innerhalb der Steuerzentrale finden Sie keine Verweise auf den Performance Monitor.

Erläuterung: Die Funktion für den Performance Monitor der Steuerzentrale wurde entfernt.

Maßnahme: Wenn Sie mit DB2 Universal Database™ (DB2 UDB) für Windows® arbeiten, stehen Tools zur Überwachung der Leistung zur Verfügung:

- **DB2 Performance Expert**

  Das getrennt erhältliche Produkt DB2 Performance Expert for Multiplatforms Version 1.1 konsolidiert Berichte, analysiert und empfiehlt Änderungen zur Optimierung der Selbstverwaltung und der Ressourcen auf der Grundlage leistungsbezogener DB2 UDB-Informationen.
- **DB2 UDB-Diagnosezentrale**

  Die Funktionen der Diagnosezentrale geben Ihnen verschiedene Methoden an die Hand, mit leistungsbezogenen Informationen zu arbeiten. Diese Funktionen bieten eine Art Ersatz für die Funktionen des Performance Monitor der Steuerzentrale.
- **Windows-Systemmonitor**

  Der Windows-Systemmonitor ermöglicht Ihnen eine Überwachung der Datenbank- und der Systemleistung, indem Sie Informationen aus beliebigen der Leistungsdatenprovider (Datenquellen) abrufen, die im System registriert sind. Windows bietet zudem Leistungsdaten zu allen Aspekten des Maschinenbetriebs, wie zum Beispiel:

  – CPU-Auslastung

– Speicherauslastung
– Plattenaktivität
– Netzwerkaktivität

Gleichzeitiges Ausführen von Onlinedienstprogrammen

Windows	UNIX

Symptom: Wenn Onlinedienstprogramme gleichzeitig verwendet werden, kann die Ausführung der Dienstprogramme lange Zeit in Anspruch nehmen.

Erläuterung: Die Sperren, die für ein Dienstprogramm erforderlich sind, beeinträchtigen den Fortschritt der anderen Dienstprogramme, die gleichzeitig ausgeführt werden.

Maßnahme: Wenn es ein Konfliktpotenzial zwischen den Sperranforderungen von Dienstprogrammen gibt, die gleichzeitig ausgeführt werden, sollten Sie eine alternative Terminierung der Dienstprogramme, die Sie ausführen wollen, in Betracht ziehen. Die Dienstprogramme (z. B. zur Onlinesicherung von Tabellenbereichen, zum Laden von Daten in eine Tabelle oder zur Inplace-Reorganisation von Tabellen) arbeiten mit Sperrmechanismen, um Konflikte zwischen den Dienstprogrammen zu vermeiden. Die Dienstprogramme verwenden Tabellensperren, Tabellenbereichssperren und Tabellenbereichsstatus zu verschiedenen Zeiten, um die Funktionen zu steuern, die in der Datenbank auszuführen sind. Wenn Sperren von einem Dienstprogramm gehalten werden, müssen andere Dienstprogramme, die ähnliche oder damit zusammenhängende Sperren anfordern, warten, bis die Sperren freigegeben werden.

Zum Beispiel kann die letzte Phase einer Inplace-Tabellenreorganisation nicht gestartet werden, solange eine Onlinesicherung ausgeführt wird, die die zu reorganisierende Tabelle mit einbezieht. Sie können die Reorganisationsanforderung anhalten, wenn die Sicherung abgeschlossen werden muss.

Ein anderes Beispiel ist das Dienstprogramm zum Onlineladen von Daten, das nicht mit einer weiteren Anforderung des Onlineladeprogramms für die gleiche Tabelle funktioniert. Wenn Daten in verschiedene Tabellen geladen werden, kommt es zu keiner gegenseitigen Sperre der Ladeanforderungen.

Änderungen an der zusammenfassenden Ausgabe von 'db2move'

Windows	UNIX

Änderung: In Version 8.2 wurde die vom Befehl **db2move** generierte Übersichtsausgabe durch geeignetere Beschreibungen verbessert. Die Änderung in der Übersichtsausgabe kann jedoch dazu führen, dass eine Prozedur, die zur Analyse der alten Ausgabe geschrieben wurde, jetzt fehlschlägt.

Symptom: Eine Prozedur, die zur Analyse der alten, von **db2move** generierten Ausgabe geschrieben wurde, schlägt fehl.

Erläuterung: Die vom Befehl **db2move** generierte Übersichtsausgabe wurde verbessert.

Bei der Ausführung des Befehls **db2move** mit der Option „IMPORT" sah die alte Ausgabe wie folgt aus:

```
IMPORT: -Rows read:     5; -Rows committed:     5;   Table "DSCIARA2"."T20"
```

Die neue Ausgabe sieht wie folgt aus:

```
* IMPORT:  table "DSCIARA2"."T20"
  -Rows read:          5
  -Inserted:           4
  -Rejected:           1
  -Committed:          5
```

Bei der Ausführung des Befehls **db2move** mit der Option „LOAD" sah die alte Ausgabe wie folgt aus:

```
* LOAD:  table "DSCIARA2"."T20"
  -Rows read:    5; -Loaded:    4;  -Rejected   1  -Deleted   0  -Committed   5
```

Die neue Ausgabe sieht wie folgt aus:

```
* IMPORT:  table "DSCIARA2"."T20"
  -Rows read:          5
  -Loaded:             4
  -Rejected:           1
  -Deleted:            0
  -Committed:          5
```

Maßnahme: Ihre Prozedur zur Analyse der Ausgabe des Befehls **db2move** muss modifiziert werden, um den Änderungen an Layout und Inhalt Rechnung zu tragen.

Änderungen an den Tabellen der EXPLAIN-Einrichtung

Windows	UNIX

Änderung: In Version 8 wurden Änderungen an den vorhandenen Tabellen der EXPLAIN-Einrichtung vorgenommen und zwei neue Tabellen hinzugefügt: ADVISE_MQT und ADVISE_PARTITION.

Symptom: Der DB2-Designadvisor gibt Fehlernachrichten zurück, wenn Empfehlungen zu gespeicherten Abfragetabellen (MQTs) oder Datenbankpartitionen angefordert werden und die EXPLAIN-Tabellen nicht erstellt wurden.

Erläuterung: Die neuen Tabellen ADVISE_MQT und ADVISE_PARTITION wurden nicht erstellt.

Maßnahme: Verwenden Sie den Befehl **db2exmig**, um die EXPLAIN-Tabellen von Version 7 und Version 8.1 in Version 8.2 zu versetzen. Dieser Befehl besitzt die erforderliche EXPLAIN-DLL-Datei, um alle erforderlichen Tabellen der EXPLAIN-Einrichtung zu erstellen.

Änderungen am Nachrichtenformat in 'db2diag.log'

Windows	UNIX

Änderung: In Version 8 wurde das Nachrichtenformat in 'db2diag.log' geändert.

Symptom: Sie werden bemerken, dass das Format geändert wurden, wenn Sie die Nachrichten in der Datei 'db2diag.log' prüfen. Die Änderungen betreffen zum Beispiel die folgenden Bereiche: Jede Nachricht hat einen Datensatzkopf des

Diagnoseprotokolls, Datensatzfeldern wird der Feldname und die Spalte vorangestellt, und die Nachrichten- und Datenabschnitte des Protokolldatensatzes werden deutlich markiert. Die Änderungen am Format machen den Protokolldatensatz insgesamt benutzerfreundlicher und verständlicher.

Erläuterung: Die DB2 UDB-Diagnoseprotokolle werden überarbeitet. Die Datei 'db2diag.log' wird durch einen Parser analysierbar.

Befehle CREATE DATABASE und DROP DATABASE früherer Versionen nicht unterstützt

Windows	UNIX

Änderung: In Version 8 werden die Befehle CREATE DATABASE und DROP DATABASE von Clients oder an Server früherer Versionen nicht unterstützt.

Symptom: Sie empfangen den Fehler SQL0901N, wenn Sie einen dieser Befehle absetzen.

Erläuterung: Die Befehle CREATE DATABASE und DROP DATABASE werden beide nur von Clients der Version 8 an Server der Version 8 unterstützt. Sie können diese Befehle nicht von einem Client der Versionen 6 oder 7 an einen Server der Version 8 absetzen. Sie können diese Befehle nicht von einem Client der Version 8 an einen Server der Version 7 absetzen.

Maßnahme: Erstellen oder Löschen Sie eine Datenbank der Version 8 von einem Client der Version 8. Erstellen oder Löschen Sie eine Datenbank der Version 7 von einem Client der Version 6 oder 7.

Modusänderung an Tabellen nach LOAD

Windows	UNIX

Änderung: In früheren Versionen war eine Tabelle, die mit der Option INSERT geladen wurde und die unmittelbare gespeicherte Abfragetabellen (auch als Übersichtstabellen bezeichnet) hatte, im Status 'Normal' (Vollzugriff), nachdem eine nachfolgende Anweisung SET INTEGRITY IMMEDIATE CHECKED für sie ausgeführt wurde. In Version 8 befindet sich die Tabelle im Modus 'Kein Versetzen von Daten', nachdem die Anweisung SET INTEGRITY IMMEDIATE CHECKED ausgeführt wurde.

Erläuterung: Der Zugriff auf eine Tabelle im Modus 'Kein Versetzen von Daten' ist dem Zugriff auf eine Tabelle im Modus 'Normal' (Vollzugriff) sehr ähnlich, außer für einige Anweisungen und Dienstprogramme, die ein Versetzen von Daten innerhalb der Tabelle bewirken.

Maßnahme: Sie können die Basistabelle, in die Daten geladen wurden und die unmittelbare Übersichtstabellen hat, veranlassen, den Modus 'Kein Versetzen von Daten' zu umgehen und direkt in den Modus 'Vollzugriff' zu wechseln, indem Sie eine Anweisung SET INTEGRITY ... IMMEDIATE CHECKED FULL ACCESS für die Basistabelle absetzen. Allerdings wird die Verwendung dieser Option nicht empfohlen, da sie eine vollständige Aktualisierung der abhängigen unmittelbaren gespeicherten Abfragetabelle (auch als Übersichtstabelle bezeichnet) erzwingt.

Dienstprogramm LOAD im Einfüge- und Ersetzungsmodus

Windows	UNIX

Änderung: Wenn in früheren Versionen das Dienstprogramm LOAD im Einfüge- oder Ersetzungsmodus verwendet wurde, hieß bei inaktivierter Integritätsprüfung die Standardoption CASCADE IMMEDIATE. Wenn die Tabelle in den Status 'Überprüfung anstehend' versetzt wurde, wurden alle Tabellen mit von ihr abhängigen Fremdschlüsseln und abhängige gespeicherte Tabellen (auch als Übersichtstabelle bezeichnet) ebenfalls sofort in den Status 'Überprüfung anstehend' versetzt.

Für Version 8 gilt bei Verwendung des Dienstprogramms LOAD im Einfüge- oder Ersetzungsmodus die Standardoption CASCADE DEFERRED, wenn die Integritätsprüfung inaktiviert ist.

Maßnahme: Sie können Tabellen mit abhängigen Fremdschlüsseln und abhängige gespeicherte Abfragetabellen zusammen mit ihren übergeordneten Tabellen in den Status 'Überprüfung anstehend' versetzen, indem Sie die Option CHECK PENDING CASCADE IMMEDIATE des Befehls LOAD angeben.

DB2_LIKE_VARCHAR steuert die Erfassung von Statistiken für Unterelemente nicht

Windows	UNIX

Änderung: In Version 7 diente die Registriervariable DB2_LIKE_VARCHAR zur Steuerung der Erfassung von Unterelementstatistiken und der Verwendung dieser Statistiken. In Version 8 steuert DB2_LIKE_VARCHAR die Erfassung von Statistiken zu Unterelementen nicht mehr, sondern die Erfassung von Unterelementstatistiken wird durch die Option LIKE STATISTICS des Befehls RUNSTATS oder durch den Wert DB2RUNSTATS_COLUMN_LIKE_STATS im Parameter iColumnflags des API db2Runstats gesteuert.

Symptom: Nach dem Aufruf des Befehls RUNSTATS oder des API db2Runstats wird die Unterelementstatistik auf den Wert -1 (Standardwert) im Systemkatalog gesetzt. Dies kann durch eine Abfrage wie die folgende geprüft werden:

```
SELECT SUBSTR(TABSCHEMA,1,18), SUBSTR(TABNAME,1,18),
   SUBSTR(COLNAME,1,18), COLCARD, AVGCOLLEN, SUB_COUNT, SUB_DELIM_LENGTH
   FROM SYSSTAT.COLUMNS
   WHERE COLNAME IN ('P_TYPE', 'P_NAME')
   ORDER BY 1,2,3
```

(Ersetzen Sie P_TYPE und P_NAME durch die entsprechenden Spaltennamen.)

Wenn das Ergebnis für eine Spalte einen nicht negativen Wert für COLCARD und AVGCOLLEN hat, jedoch den Wert -1 für SUB_COUNT und SUB_DELIM-_LENGTH, weist dies darauf hin, dass die Basisstatistiken für die Spalte erfasst wurden, jedoch die Unterelementstatistiken nicht erfasst wurden.

Maßnahme: Wenn Sie DB2_LIKE_VARCHAR=?,Y (wobei ? ein beliebiger Wert ist) in Version 7 angegeben haben, sollten Sie die Option LIKE STATISTICS im Befehl RUNSTATS oder den Wert DB2RUNSTATS_COLUMN_LIKE_STATS im API db2Runstats angeben, um diese Statistiken für die entsprechenden Spalten zu erfassen.

Konnektivität und Koexistenz

Unterstützung für Server früherer Versionen

Windows	UNIX

Änderung: Wenn Sie bei der Umstellung Ihrer Umgebung von Version 7 auf Version 8 in die Situation kommen, dass Sie Ihre Clientmaschinen auf Version 8 migrieren, bevor Sie alle Ihre Server auf Version 8 migrieren, müssen Sie verschiedene Einschränkungen und Ausschlüsse beachten. Diese Einschränkungen und Ausschlüsse sind weder mit DB2 Connect noch mit zSeries-, OS/390- oder iSeries-Datenbankservern verbunden.

Maßnahme: Damit Clients der Version 8 mit Servern der Version 7 arbeiten können, müssen Sie die Verwendung der DRDA-Anwendungsserverfunktion auf dem Server der Version 7 konfigurieren und aktivieren. Informationen dazu finden Sie in *Installation und Konfiguration Ergänzung* der Version 7.

Um die bekannten Einschränkungen und Ausschlüsse zu vermeiden, sollten Sie alle Ihre Server auf Version 8 migrieren, bevor Sie eine der Clientmaschinen auf Version 8 migrieren. Wenn dies nicht möglich ist, sollten Sie wissen, dass beim Zugriff auf Server der Version 7 von Clients der Version 8 keine Unterstützung für folgende Elemente verfügbar ist:

- Einige Datentypen:
  - Datentypen für große Objekte (LOB)
  - Benutzerdefinierte einzigartige Datentypen (UDTs)
  - DATALINK-Datentypen
- Einige Sicherheitsfunktionen:
  - Authentifizierungstyp SERVER_ENCRYPT
  - Ändern von Kennwörtern Sie können von einem Client der DB2 UDB Version 8 aus keine Kennwörter auf einem Server der DB2 UDB Version 7 ändern.
- Bestimmte Verbindungen und Kommunikationsprotokolle:
  - Exemplaranforderungen, für eine ATTACH-Verbindung anstatt einer CONNECT-Verbindung erforderlich ist
  - Die Anweisung ATTACH wird von einem Client der DB2 UDB Version 8 an einen Server der DB2 UDB Version 7 nicht unterstützt.
  - Das einzig unterstützte Netzwerkprotokoll ist TCP/IP.
  - Andere Netzwerkprotokolle wie SNA, NetBIOS, IPX/SPX und andere werden nicht unterstützt.
- Einige Anwendungsfunktionen und Tasks:
  - Die Anweisung DESCRIBE INPUT wird, mit einer Ausnahme für ODBC/ JDBC-Anwendungen, nicht unterstützt. Zur Unterstützung von Clients der DB2 UDB Version 8, die ODBC/ JDBC-Anwendungen ausführen, die auf Server der DB2 UDB Version 7 zugreifen, muss eine Korrektur (Fix) für die DESCRIBE INPUT-Unterstützung auf alle Server der DB2 UDB Version 7 angewendet werden, auf die diese Art von Zugriff erforderlich ist. Diese Korrektur ist mit APAR IY30655 verbunden und wird vor dem Datum der allgemeinen Veröffentlichung von DB2 UDB Version 8 (General Availability) verfügbar sein. Informationen dazu, wie Sie die Korrektur mit APAR IY30655

erhalten, finden Sie im Abschnitt „Kontaktaufnahme mit IBM" in einem beliebigen DB2 UDB-Dokument. Die Anweisung DESCRIBE INPUT ist eine Erweiterung zu besserer Leistung und Benutzerfreundlichkeit, die einem Anwendungsrequester die Möglichkeit gibt, eine Beschreibung von Eingabeparametermarken in einer vorbereiteten (PREPARE) Anweisung abzurufen. Bei einer Anweisung CALL umfasst dies auch die Parametermarken, die den Parametern IN und INOUT für eine gespeicherte Prozedur zugeordnet sind.

- Wenn Sie den Befehl Result.getObject(1) verwenden, wird der Datentyp BigDecimal an Stelle des Datentyps Java Long zurückgegeben, wie durch die JDBC-Spezifikation angefordert. Der DB2 UDB Version 7 DRDA-Server ordnet BIGINT zu DEC(19,0) wenn er auf eine Anfrage DESCRIBE INPUT antwortet und wenn er Daten abruft. Diese Verhalten tritt auf, weil der DB2 UDB Version 7-Server auf einer DRDA-Stufe arbeitet, bei der BIGINT nicht definiert ist.
- Abfrageunterbrechungen werden nicht unterstützt. Diese betrifft das Verbindungsattribut CLI/ODBC SQL_QUERY_TIMEOUT sowie die Unterbrechungs-APIs.
- Zweiphasige Festschreibung. Ein Server der DB2 UDB Version 7 kann nicht als Transaktionsmanagerdatenbank fungieren, wenn koordinierte Transaktionen ausgeführt werden, an denen Clients der DB2 UDB Version 8 beteiligt sind. Ein Server der DB2 UDB Version 7 kann auch nicht an einer koordinierten Transaktion teilnehmen, in der ein Server der DB2 UDB Version 8 möglicherweise als Transaktionsmanagerdatenbank fungiert.
- XA-konforme Transaktionsmanager. Eine Anwendung, die einen Client der DB2 UDB Version 8 verwendet, kann keinen Server der DB2 UDB Version 7 als XA-Ressource nutzen. Dies schließt WebSphere, Microsoft COM+/MTS, BEA WebLogic und andere mit ein, die Teil einer Anordnung zur Transaktionsverwaltung sind.
- Überwachung. Überwachungsfunktionen werden von einem Client der DB2 UDB Version 8 an einen Server der DB2 UDB Version 7 nicht unterstützt.
- Dienstprogramme. Die Dienstprogramme, die von einem Client auf einem Server gestartet werden können werden nicht unterstützt, wenn:
  1. Der Client ist auf DB2 UDB Version 8 und der Server ist auf DB2 UDB Version 7.
  2. SQL-Anweisungen, die größer als 32 KB sind.
- Abfrageunterbrechungen werden nicht unterstützt. Diese betrifft das Verbindungsattribut CLI/ODBC SQL_QUERY_TIMEOUT sowie die Unterbrechungs-APIs.

Neben diesen Einschränkungen und Ausschlüssen für Clients der DB2 UDB Version 8, die mit Servern der DB2 UDB Version 7 arbeiten, gibt es ähnliche Einschränkungen und Ausschlüsse auch für Tools der DB2 UDB Version 8, die mit Servern der DB2 UDB Version 7 arbeiten. Die folgenden Tools der DB2 UDB Version 8 unterstützen lediglich Server der DB2 UDB Version 8:

- Steuerzentrale
- Taskzentrale
- Journal
- Satellitenverwaltungszentrale
- Informationskatalogzentrale (einschließlich der Webversion dieser Zentrale)

- Diagnosezentrale (einschließlich der Webversion dieser Zentrale)
- Lizenzzentrale
- Spatial Extender
- Toolseinstellungen
- Entwicklungszentrale. Verwenden Sie Stored Procedure Builder, um Serverobjekte auf Servern zu entwickeln, die älter als Version 8 sind.

Die folgenden Tools der DB2 UDB Version 8 unterstützen Server der DB2 UDB Version 7 (mit einigen Einschränkungen) und Server der DB2 UDB Version 8:

- Konfigurationsassistent

  Es ist möglich, einen Server der DB2 UDB Version 7 aufzuspüren und diesen zu katalogisieren. Wenn Sie versuchen, auf den Server der DB2 UDB Version 7 zuzugreifen, werden keine Funktionen arbeiten obwohl er katalogisiert ist. Außerdem können Sie ein Profil der DB2 UDB Version 7 in einen Server der DB2 UDB Version 8 importieren oder ein Profil der DB2 UDB Version 8 in einen Server der DB2 UDB Version 7 importieren. Alle anderen Funktionen des Konfigurationsassistenten werden mit Servern der DB2 UDB Version 7 nicht arbeiten.
- Data Warehouse-Zentrale
- Replikationszentrale
- Befehlseditor (der Ersatz für die Befehlszentrale, einschließlich der Webversion dieser Zentrale)

  Das Importieren und Speichern von Prozeduren auf und von Servern der DB2 UDB Version 7 ist nicht möglich. Dienstprogramme, die ATTACH erfordern, können nicht verwendet werden.
- SQL Assist
- Visual Explain

Allgemein gilt, dass jedes Tool der DB2 UDB Version 8, das nur aus der Navigationsstruktur der Steuerzentrale heraus gestartet wird, bzw. alle Detailsichten aus solchen Tools, nicht für Server der DB2 UDB Version 7 und früheren Versionen verfügbar bzw. zugänglich sind. Sie sollten eine Verwendung der Tools von DB2 UDB Version 7 in Betracht ziehen, wenn Sie mit Servern der DB2 UDB Version 7 oder früheren Versionen arbeiten.

Unterstützung für verschiebbare Cursor

Windows	UNIX

Änderung: In Version 8 wird die Funktionalität verschiebbarer Cursor von einem Client mit DB2 UDB für Unix und Windows der Version 8 zu einem Server mit DB2 UDB für Unix und Windows der Version 7 nicht unterstützt. Die Unterstützung für verschiebbare Cursor steht nur von einem Client mit DB2 UDB für Unix und Windows der Version 8 zu einem Server mit DB2 UDB für Unix und Windows der Version 8 oder zu einem Server mit DB2 UDB für z/OS und OS/390 der Version 7 zur Verfügung. Clients mit DB2 UDB für Unix und Windows der Version 7 unterstützen die vorhandene Funktionalität verschiebbarer Cursor zu Servern mit DB2 UDB für Unix und Windows der Version 8 weiterhin.

Maßnahme: Führen Sie einen Upgrade Ihrer Server auf Version 8 durch.

Zugriff auf Server der Version 7 über einen DB2 Connect-Server der Version 8

Windows	UNIX

Änderung: In Version 8 wird der Zugriff von einem Client mit DB2 UDB für Unix und Windows zu einem Server mit DB2 UDB der Version 7 über ein Server der Version 8 nicht unterstützt, wenn die Funktionalität durch DB2 Connect Enterprise Edition Version 8 oder DB2 UDB Enterprise Server Edition Version 8 bereitgestellt wird.

Maßnahme: Führen Sie einen Upgrade Ihrer Server auf Version 8 durch.

Typ-1-Verbindung über den Befehlszeilenprozessor und eingebettetes SQL

Windows	UNIX

Änderung: Wenn Sie in früheren Versionen von DB2 UDB den Befehlszeilenprozessor (CLP) oder eingebettetes SQL und eine Typ-1-Verbindung zu einer Datenbank verwendeten, schlug der Versuch, während einer Arbeitseinheit eine Verbindung zu einer anderen Datenbank herzustellen, mit einem Fehler SQL0752N fehl. In Version 8 wird die Arbeitseinheit festgeschrieben, die Verbindung wird zurückgesetzt und die Verbindung zu der zweiten Datenbank wird zugelassen. Das Festschreiben der Arbeitseinheit und das Zurücksetzen der Verbindung erfolgen auch, wenn AUTOCOMMIT inaktiviert ist.

Nachrichten

DB2 Connect-Nachrichten werden anstelle von DB2 UDB-Nachrichten zurückgegeben

Windows	UNIX

Änderung: In Version 8 können Bedingungen, die in früheren Releases eine DB2 UDB-Nachricht zurückgegeben hätten, nun eine DB2 Connect-Nachricht zurückgeben.

Die von dieser Änderung betroffenen Nachrichten beziehen sich auf Binde-, Verbindungs- und Sicherheitsfehler. SQL-Fehler für Abfragen und andere SQL-Anforderungen sind von dieser Änderungen nicht betroffen.

Beispiele:

- SQLCODE -30081 wird anstelle von SQLCODE -1224 zurückgegeben.
- SQLCODE -30082 wird anstelle von SQLCODE -1403 zurückgegeben.
- SQLCODE -30104 wird anstelle von SQLCODE -4930 zurückgegeben.

Symptom: Anwendungen, deren Code eine Reaktion auf DB2 UDB-Nachrichten vorsieht, verhalten sich möglicherweise nicht mehr wie zuvor.

Konfigurationsparameter

Veraltete Konfigurationsparameter des Datenbankmanagers

Windows	UNIX

Änderung: Die folgenden Konfigurationsparameter der Datenbankmanagers sind veraltet:

- *backbufsz*: In früheren Versionen konnten Sie eine Sicherungsoperation mit einer Standardpuffergröße durchführen, wobei der Wert *backbufsz* als Standardwert genommen wurde. In Version 8 sollten Sie die Größe Ihrer Sicherungspuffer explizit angeben, wenn Sie das Sicherungsdienstprogramm verwenden.
- *dft_client_adpt*: DCE Verzeichnisservices werden nicht länger unterstützt.
- *dft_client_comm*: DCE Verzeichnisservices werden nicht länger unterstützt.
- *dir_obj_name*: DCE Verzeichnisservices werden nicht länger unterstützt.
- *dir_path_name*: DCE Verzeichnisservices werden nicht länger unterstützt.
- *dir_type*: DCE Verzeichnisservices werden nicht länger unterstützt.
- *dos_rqrioblk*
- *drda_heap_sz*
- *fcm_num_anchors*, *fcm_num_connect* und *fcm_num_rqb*: DB2 UDB passt Nachrichtenanker, Verbindungseinträge und Anforderungsblöcke dynamisch und automatisch an, so dass Sie diese Parameter nicht mehr definieren müssen.
- *fileserver*: IPX/SPX wird nicht länger unterstützt.
- *initdari_jvm*: Gespeicherte Java-Prozeduren werden nun standardmäßig als Multithreadprozeduren ausgeführt. Sie werden in von Routinen anderer Sprachen getrennten Prozessen ausgeführt, so dass dieser Parameter nicht länger unterstützt wird.
- *ipx_socket*: IPX/SPX wird nicht länger unterstützt.
- *jdk11_path*: Wurde durch den Konfigurationsparameter *jdk_path* des Datenbankmanagers ersetzt.
- *keepdari*: Wurde durch den Konfigurationsparameter *keepfenced* des Datenbankmanagers ersetzt.
- *max_logicagents*: Wurde durch den Konfigurationsparameter *max_connections* des Datenbankmanagers ersetzt.
- *maxdari*: Wurde durch den Konfigurationsparameter *fenced_pool* des Datenbankmanagers ersetzt.
- *num_initdaris*: Wurde durch den Konfigurationsparameter *num_initfenced* des Datenbankmanagers ersetzt.
- *objectname*: IPX/SPX wird nicht länger unterstützt.
- *restbufsz*: In früheren Versionen konnten Sie eine Wiederherstellungsoperation (RESTORE) mit einer Standardpuffergröße durchführen, wobei der Wert *restbufsz* als Standardwert genommen wurde. In Version 8 sollten Sie die Größe Ihrer Wiederherstellungspuffer explizit angeben, wenn Sie das Wiederherstellungsdienstprogramm verwenden.
- *route_obj_name*: DCE Verzeichnisservices werden nicht länger unterstützt.
- *ss_logon*: Dies ist ein OS/2-Parameter, und OS/2 wird nicht länger unterstützt.
- *udf_mem_sz*: Benutzerdefinierte Funktionen (UDFs) übergeben keine Daten im gemeinsamen Speicher mehr, so dass dieser Parameter nicht mehr unterstützt wird.

Maßnahme: Entfernen Sie alle Verweise auf diese Parameter aus Ihren Anwendungen.

Veraltete Datenbankkonfigurationsparameter

Windows	UNIX

Änderung: Die folgenden Datenbankkonfigurationsparameter sind veraltet:

- *buffpage*: In früheren Versionen konnten Sie einen Pufferpool mit einer Standardgröße erstellen oder ändern, wobei der Wert des Parameters *buffpage* als Standardwert genommen wurde.In Version 8 sollten Sie die Größe Ihrer Pufferpools explizit mit Hilfe des Schlüsselworts SIZE in den Anweisungen ALTER BUFFERPOOL oder CREATE BUFFERPOOL angeben.
- *copyprotect*
- *indexsort*

Maßnahme: Entfernen Sie alle Verweise auf diese Parameter aus Ihren Anwendungen.

Inkompatibilitäten von Version 7 zu vorigen Releases

Anwendungsprogrammierung

Query Patroller Universal Client

WIN	UNIX	OS/2

Änderung: Diese neue Version des Client Application Enabler (CAE) funktioniert nur mit Query Patroller Server Version 7, weil neue gespeicherte Prozeduren vorhanden sind. CAE ist die Anwendungsschnittstelle für DB2 Universal Database™ (DB2 UDB), die letzten Endes alle Anwendungen passieren müssen, um auf die Datenbank zuzugreifen.

Symptom: Wenn diese CAE-Version mit einem Server einer früheren Version ausgeführt wird, wird SQL29001 zurückgegeben.

Objektumsetzungsfunktionen und strukturierte Typen

WIN	UNIX	OS/2

Änderung: Es gibt eine kleinere und entfernt mögliche Inkompatibilität zwischen einem Client einer Version vor Version 7 und einem Server der Version 7, die im Zusammenhang mit Änderungen am SQL-Deskriptorbereich (SQLDA) stehen. Byte 8 des zweiten SQLVAR-Feldes kann nun den Wert X'12' (neben den Werten X'00' und X'01') annehmen. Anwendungen, die den neuen Wert nicht abfangen, können von dieser Erweiterung beeinträchtigt werden.

Maßnahme: Da es in zukünftigen Releases noch weitere Erweiterungen an diesem Feld geben kann, sind Entwickler gut beraten, dieses Feld nur auf explizit definierte Werte zu testen.

Von JVM verwendete Versionen von Class- und Jar-Dateien

WIN	UNIX	OS/2

Änderung: Wenn früher eine gespeicherte Java-Prozedur oder eine benutzerdefinierte Funktion (UDF) gestartet wurde, sperrte Java Virtual Machine (JVM) alle Dateien, die in CLASSPATH angegeben waren (einschließlich der Dateien in `sqllib/function`). JVM verwendete diese Dateien, bis der Datenbankmanager gestoppt wurde. Abhängig von der Umgebung, in der Sie eine gespeicherte Prozedur oder UDF ausführen (d. h. abhängig vom Wert des Konfigurationsparameters *keepdari* des Datenbankmanagers und davon, ob die gespeicherte Prozedur abgeschirmt (fenced) wird), ermöglicht Ihnen das Aktualisieren (Refresh) von Klassen ein Ersetzen von CLASS- und JAR-Dateien, ohne den Datenbankmanager zu stoppen. Dies unterscheidet sich von der früheren Funktionsweise.

Geänderte Funktionalität der Befehle zum Installieren, Ersetzen und Entfernen eines JAR

WIN	UNIX	OS/2

Änderung: In der Vergangenheit bewirkte die Installation eines JAR ein Abbrechen aller DARI-Prozesse (Database Application Remote Interface). Auf diese Weise wurde sichergestellt, dass eine neue gespeicherte Prozedurklasse garantiert mit dem nächsten Aufruf ausgewählt wurde. Zur Zeit brechen keine JAR-Befehle DARI-Prozesse ab. Um sicherzustellen, dass Klassen aus neu installierten oder ersetzten JARs ausgewählt werden, müssen Sie den Befehl SQLEJ.REFRESH_CLASSES explizit absetzen.

Eine weitere Inkompatibilität, die durch das Nichtabrechen von DARI-Prozessen eingeführt wird, ist die Tatsache, dass für abgeschirmte (fenced) gespeicherte Prozeduren, wenn der Wert des Konfigurationsparameters *keepdari* des Datenbankmanagers auf "YES" gesetzt ist, Clients verschiedene Versionen der JAR-Dateien erhalten können. Betrachten Sie folgendes Szenario:

1. Benutzer A ersetzt ein JAR und aktualisiert (Refresh) die Klassen nicht.
2. Benutzer A ruft anschließend eine gespeicherte Prozedur aus dem JAR auf. Unter der Annahme, dass dieser Aufruf denselben DARI-Prozess verwendet, erhält Benutzer A eine alte Version der JAR-Datei.
3. Benutzer B ruft dieselbe gespeicherte Prozedur auf. Dieser Aufruf arbeitet mit einer neuen DARI, was bedeutet, dass das neu erstellte Klassenladeprogramm (Class loader) die neue Version der JAR-Datei auswählt.

Mit anderen Worten, wenn Klassen nach JAR-Operationen nicht aktualisiert (Refresh) werden, kann eine gespeicherte Prozedur aus verschiedenen Versionen von JARs aufgerufen werden, je nachdem, welche DARI-Prozesse verwendet werden. Dies unterscheidet sich von der früheren Funktionsweise, bei der (durch Abbrechen (Flush) der DARI-Prozesse) sichergestellt war, dass neue Klassen immer verwendet wurden.

Inkompatibilität bei 32-Bit-Anwendungen

	UNIX	

Änderung: Ausführbare 32-Bit-Codedateien (DB2 UDB-Anwendungen) funktionieren mit der neuen 64-Bit-Datenbanksteuerkomponente nicht.

Symptom: Die Anwendung kann nicht verbunden (″gelinkt″) werden. Wenn Sie versuchen, 32-Bit-Objekte mit der 64-Bit-Anwendungsbibliothek von DB2 UDB zu verbinden, wird eine Betriebssystemfehlernachricht des Verbindungseditors (Linker) angezeigt.

Maßnahme: Die Anwendung muss als ausführbare 64-Bit-Codedatei neu kompiliert und wieder mit den neuen 64-Bit-Bibliotheken von DB2 UDB verbunden (gelinkt) werden.

Ändern des Längenfelds des Arbeitspuffers

WIN	UNIX	OS/2

Änderung: Jede benutzerdefinierte Funktion (UDF), die das Längenfeld des Arbeitspuffers ändert, der an die UDF übergeben wurde, empfängt nun einen SQLCODE-Wert -450.

Symptom: Eine UDF, die das Längenfeld des Arbeitspuffers ändert, schlägt fehl. Die aufrufende Anweisung empfängt den SQLCODE-Wert -450 mit Angabe des Schemas und des bestimmten Namens der Funktion.

Maßnahme: Schreiben Sie den Hauptteil der UDF um, so dass das Längenfeld des Arbeitspuffers nicht geändert wird.

SQL

Anwendungen, die mit dem Schema SESSION qualifizierte reguläre Tabellen verwenden

WIN	UNIX	OS/2

Änderung: Das Schema SESSION ist das einzig zulässige Schema für temporäre Tabellen und wird nun von DB2 UDB verwendet, um anzuzeigen, dass ein mit SESSION qualifizierter Tabellenname auf eine temporäre Tabelle verweisen kann. Jedoch ist SESSION kein reserviertes Schlüsselwort für temporäre Tabellen und kann als Schema für reguläre Basistabellen verwendet werden. Aus diesem Grund kann eine Anwendung vielleicht sowohl eine richtige Tabelle SESSION.T1 als auch eine deklarierte temporäre Tabelle SESSION.T1 gleichzeitig vorfinden. Wenn beim Binden eines Pakets eine statische Anweisung angetroffen wird, die einen (explizit oder implizit) durch SESSION qualifizierten Tabellenverweis enthält, wird weder ein Abschnitt (Section) noch eine Abhängigkeit für diese Anweisung in den Katalogen gespeichert. Statt dessen muss dieser Abschnitt zur Laufzeit inkrementell gebunden werden. Dadurch wird eine Kopie des Abschnitts in den Cache für dynamisches SQL gestellt, wo die zwischengespeicherte Kopie nur für das eindeutige Exemplar der Anwendung privat ist. Wenn zur Laufzeit eine deklarierte temporäre Tabelle mit übereinstimmendem Tabellennamen vorhanden ist, wird die deklarierte temporäre Tabelle verwendet, selbst wenn eine permanente Basistabelle desselben Namens existiert.

Symptom: In Version 6 (und früheren) verweisen alle Pakete mit statischen Anweisungen, die Tabellen mit durch SESSION qualifizierten Namen betreffen, immer auf eine permanente Basistabelle. Beim Binden eines Pakets werden ein Abschnitt (Section) sowie relevante Abhängigkeitseinträge für diese Anweisung in den Katalogen gespeichert. In Version 7 werden diese Anweisungen nicht zur Bindezeit gebunden und können zur Laufzeit möglicherweise in eine deklarierte temporäre Tabelle des gleichen Namens aufgelöst werden. Daraus können folgende Situationen entstehen:

- Bei Migration von Version 5. Wenn ein solches Paket in Version 5 vorhanden war, wird es in Version 6 erneut gebunden, und die statischen Anweisungen werden nun inkrementell gebunden. Dies kann sich auf die Leistung auswirken, da diese inkrementell gebundenen Abschnitte (Sections) sich wie im Cache zwischengespeichertes dynamisches SQL verhalten, mit der Ausnahme, dass der im Cache zwischengespeicherte Abschnitt nicht mit anderen Anwendungen gemeinsam benutzt werden kann (nicht einmal von verschiedenen Exemplaren derselben ausführbaren Anwendungsdatei).
- Bei Migration von Version 6 zu Version 7. Wenn ein solches Paket in Version 6 vorhanden war, wird es in Version 7 nicht unbedingt erneut gebunden. Statt dessen werden die Anweisungen weiterhin als reguläres statisches SQL unter Verwendung des Abschnitts (Section) ausgeführt, der im Katalog zu ursprünglichen Bindezeit gespeichert wurde. Wenn dieses Paket jedoch erneut gebunden wird (implizit oder explizit), werden die Anweisungen im Paket mit durch SESSION qualifizierten Tabellenverweisen nicht mehr gespeichert und erfordern inkrementelles Binden. Dadurch kann die Leistung beeinträchtigt werden.

Zusammengefasst lässt sich sagen, dass jedes Paket, dass in Version 7 mit statischen Anweisungen gebunden wird, die auf durch SESSION qualifizierte Tabellen verweisen, sich nicht länger wie statisches SQL verhalten, weil sie ein inkrementelles Binden erfordern. Wenn der Anwendungsprozess eine Anweisung DECLARE GLOBAL TEMPORARY TABLE für eine Tabelle absetzt, die den gleichen Namen wie eine vorhandene, mit SESSION qualifizierte Tabelle, Sicht bzw. ein solcher Aliasname besitzt, werden Verweise auf diese Objekte immer auf die deklarierte temporäre Tabelle bezogen.

Maßnahme: Ändern Sie nach Möglichkeit die Schemanamen permanenter Tabellen, so dass nicht SESSION verwendet wird. Ansonsten gibt es keine Abhilfe, außer sich die Auswirkungen auf die Leistung und den möglichen Konflikt mit deklarierten temporären Tabellen bewusst zu machen.

Die folgende Abfrage kann zur Erkennung von Tabellen, Sichten und Aliasnamen verwendet werden, die betroffen sein könnten, wenn eine Anwendung mit temporären Tabellen arbeitet:

```
select tabschema, tabname from SYSCAT.TABLES where tabschema = 'SESSION'
```

Die folgende Abfrage kann zur Ermittlung von gebundenen Paketen der Version 7 dienen, für die statische Abschnitte (Sections) in den Katalogen gespeichert sind und deren Funktionsweise sich ändern könnte, wenn das Paket erneut gebunden wird (nur bei Migration von Version 6 zu Version 7 relevant):

```
select pkgschema, pkgname, bschema, bname from syscat.packagedep
   where bschema = 'SESSION' and btype in ('T', 'V', 'I')
```

Dienstprogramme und Tools

db2set unter AIX und Solaris

	UNIX	

Änderung: Der Befehl ″db2set -ul (user level)″ und die zugehörigen Funktionen wurden nicht auf AIX oder Solaris portiert.

Konnektivität und Koexistenz

32-Bit-Client-Inkompatibilität

WIN	UNIX	OS/2

Änderung: 32-Bit-Clients können keine Verbindung zu Exemplaren (Attach) oder Datenbanken (Connect) auf 64-Bit-Servern herstellen.

Symptom: Wenn sowohl der Client als auch der Server mit Code der Version 7 arbeiten, wird SQL1434N zurückgegeben. Ansonsten schlägt der Verbindungsversuch (Attach oder Connect) mit SQLCODE-Wert -30081 fehl.

Maßnahme: Verwenden Sie 64-Bit-Clients.

Anhang B. Unterstützung von Landessprachen (NLS)

Dieser Abschnitt enthält Informationen zur Unterstützung von Landessprachen (National Language Support, NLS), die von DB2 Universal Database™ (DB2 UDB) bereitgestellt wird, einschließlich Informationen zu den unterstützten Gebieten, Sprachen und Codepages (codierten Zeichensätzen) sowie Angaben zu Konfiguration und Verwendung von DB2 UDB NLS-Funktionen in Datenbanken und Anwendungen.

Versionen in anderen Nationalsprachen

DB2 Universal Database™ (DB2 UDB) Version 8 ist in folgenden Sprachen verfügbar: vereinfachtes Chinesisch, traditionelles Chinesisch, Dänisch, Deutsch, Englisch, Finnisch, Französisch, Italienisch, Japanisch, Koreanisch, Norwegisch, Polnisch, brasilianisches Portugiesisch, Russisch, Schwedisch, Spanisch und Tschechisch.

DB2 UDB Run-Time Client ist in folgenden Sprachen verfügbar: Arabisch, Bulgarisch, Griechisch, Hebräisch, Kroatisch, Niederländisch, Portugiesisch, Rumänisch, Slowakisch, Slowenisch, Türkisch und Ungarisch.

Zugehörige Referenzen:

• „Unterstützte Gebietscodes und Codepages" auf Seite 265

Unterstützte Gebietscodes und Codepages

Die folgenden Tabellen zeigen die von den Datenbankservern unterstützten Sprachen und codierten Zeichensätze sowie die Zuordnung dieser Werte zu den vom Datenbankmanager verwendeten Werten für den Gebietscode und die Codepage.

Es folgt eine Erläuterung zu den Spalten in den Tabellen:

- Unter **Codepage** wird die von IBM definierte Codepage gezeigt, wie sie vom codierten Zeichensatz des Betriebssystems zugeordnet wird.
- Unter **Gruppe** wird angegeben, ob eine Codepage Einzelbytezeichen („S") oder Doppelbytezeichen („D") enthält bzw. in dieser Hinsicht neutral („N") ist. Die Angabe „-n" ist eine Zahl, die zur Erstellung einer Buchstabe-Zahl-Kombination verwendet wird. Übereinstimmende Kombinationen geben an, wo die Verbindung und die Umsetzung von DB2 Universal Database™ (DB2 UDB) zugelassen wird. Beispielsweise können alle „S-1"-Gruppen zusammenarbeiten. Wenn die Gruppe hingegen neutral ist, sind Verbindungen und Umsetzungen mit allen anderen aufgeführten Codepages zulässig.
- Unter **Codierter Zeichens.** wird der codierte Zeichensatz für die unterstützte Sprache aufgeführt. Der codierte Zeichensatz wird der DB2 UDB-Codepage zugeordnet.
- Unter **Gebietscode** finden Sie den Code, den der Datenbankmanager intern zum Bereitstellen der regionenspezifischen Unterstützung verwendet.
- Unter **Ländereinstellung** werden die Werte der vom Datenbankmanager unterstützten länderspezifischen Angaben angezeigt.
- Unter **Betriebssystem** wird das Betriebssystem angezeigt, das die Sprachen und codierten Zeichensätze unterstützt.

Tabelle 38. Unicode

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
1200	N-1	16-Bit-Uni-code	Beliebig	Beliebig	Beliebig
1208	N-1	UTF-8-Co-dierung für Unicode	Beliebig	Beliebig	Beliebig

Tabelle 39. Albanien, Gebietskennung: AL

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
819	S-1	ISO8859-1	355	sq_AL	AIX
850	S-1	IBM-850	355	-	AIX
923	S-1	ISO8859-15	355	sq_AL.8859-15	AIX
1208	N-1	UTF-8	355	SQ_AL	AIX
37	S-1	IBM-37	355	-	Host
1140	S-1	IBM-1140	355	-	Host
819	S-1	iso88591	355	-	HP-UX
923	S-1	iso885915	355	-	HP-UX
1051	S-1	roman8	355	-	HP-UX
437	S-1	IBM-437	355	-	OS/2
850	S-1	IBM-850	355	-	OS/2
819	S-1	ISO8859-1	355	-	Solaris
923	S-1	ISO8859-15	355	-	Solaris
1252	S-1	1252	355	-	Windows

Tabelle 40. Arabische Länder/Regionen, Gebietskennung: AA

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
1046	S-6	IBM-1046	785	Ar_AA	AIX
1089	S-6	ISO8859-6	785	ar_AA	AIX
1208	N-1	UTF-8	785	AR_AA	AIX
420	S-6	IBM-420	785	-	Host
425	S-6	IBM-425	785	-	Host
1089	S-6	iso88596	785	ar_SA.iso88596	HP-UX
864	S-6	IBM-864	785	-	OS/2
1256	S-6	1256	785	-	Windows

Tabelle 41. Australien, Gebietskennung: AU

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
819	S-1	ISO8859-1	61	en_AU	AIX
850	S-1	IBM-850	61	-	AIX
923	S-1	ISO8859-15	61	en_AU.8859-15	AIX
1208	N-1	UTF-8	61	EN_AU	AIX
37	S-1	IBM-37	61	-	Host
1140	S-1	IBM-1140	61	-	Host
819	S-1	iso88591	61	-	HP-UX
923	S-1	iso885915	61	-	HP-UX
1051	S-1	roman8	61	-	HP-UX
437	S-1	IBM-437	61	-	OS/2

Tabelle 41. Australien, Gebietskennung: AU (Forts.)

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
850	S-1	IBM-850	61	-	OS/2
819	S-1	ISO8859-1	61	en_AU	SCO
819	S-1	ISO8859-1	61	en_AU	Solaris
923	S-1	ISO8859-15	61	-	Solaris
1252	S-1	1252	61	-	Windows

Tabelle 42. Österreich, Gebietskennung: AT

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
819	S-1	ISO8859-1	43	-	AIX
850	S-1	IBM-850	43	-	AIX
923	S-1	ISO8859-15	43	-	AIX
1208	N-1	UTF-8	43	-	AIX
37	S-1	IBM-37	43	-	Host
1140	S-1	IBM-1140	43	-	Host
819	S-1	iso88591	43	-	HP-UX
923	S-1	iso885915	43	-	HP-UX
1051	S-1	roman8	43	-	HP-UX
819	S-1	ISO-8859-1	43	de_AT	Linux
923	S-1	ISO-8859-15	43	de_AT@euro	Linux
437	S-1	IBM-437	43	-	OS/2
850	S-1	IBM-850	43	-	OS/2
819	S-1	ISO8859-1	43	de_AT	SCO
819	S-1	ISO8859-1	43	de_AT	Solaris
923	S-1	ISO8859-15	43	de_AT.ISO8859-15	Solaris
1252	S-1	1252	43	-	Windows

Tabelle 43. Weißrussland, Gebietskennung: BY

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
1167	S-5	KOI8-RU	375	–	–
915	S-5	ISO8859-5	375	be_BY	AIX
1208	N-1	UTF-8	375	BE_BY	AIX
1025	S-5	IBM-1025	375	-	Host
1154	S-5	IBM-1154	375	-	Host
915	S-5	ISO8859-5	375	-	OS/2
1131	S-5	IBM-1131	375	-	OS/2
1251	S-5	1251	375	-	Windows

Tabelle 44. Belgien, Gebietskennung: BE

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
819	S-1	ISO8859-1	32	fr_BE	AIX
819	S-1	ISO8859-1	32	nl_BE	AIX
850	S-1	IBM-850	32	Fr_BE	AIX
850	S-1	IBM-850	32	Nl_BE	AIX
923	S-1	ISO8859-15	32	fr_BE.8859-15	AIX
923	S-1	ISO8859-15	32	nl_BE.8859-15	AIX
1208	N-1	UTF-8	32	FR_BE	AIX

Tabelle 44. Belgien, Gebietskennung: BE (Forts.)

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
1208	N-1	UTF-8	32	NL_BE	AIX
274	S-1	IBM-274	32	-	Host
500	S-1	IBM-500	32	-	Host
1148	S-1	IBM-1148	32	-	Host
819	S-1	iso88591	32	-	HP-UX
923	S-1	iso885915	32	-	HP-UX
819	S-1	ISO-8859-1	32	fr_BE	Linux
819	S-1	ISO-8859-1	32	nl_BE	Linux
923	S-1	ISO-8859-15	32	fr_BE@euro	Linux
923	S-1	ISO-8859-15	32	nl_BE@euro	Linux
437	S-1	IBM-437	32	-	OS/2
850	S-1	IBM-850	32	-	OS/2
819	S-1	ISO8859-1	32	fr_BE	SCO
819	S-1	ISO8859-1	32	nl_BE	SCO
819	S-1	ISO8859-1	32	fr_BE	Solaris
819	S-1	ISO8859-1	32	nl_BE	Solaris
923	S-1	ISO8859-15	32	fr_BE.ISO8859-15	Solaris
923	S-1	ISO8859-15	32	nl_BE.ISO8859-15	Solaris
1252	S-1	1252	32	-	Windows

Tabelle 45. Bulgarien, Gebietskennung: BG

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
915	S-5	ISO8859-5	359	bg_BG	AIX
1208	N-1	UTF-8	359	BG_BG	AIX
1025	S-5	IBM-1025	359	-	Host
1154	S-5	IBM-1154	359	-	Host
915	S-5	iso88595	359	bg_BG.iso88595	HP-UX
855	S-5	IBM-855	359	-	OS/2
915	S-5	ISO8859-5	359	-	OS/2
1251	S-5	1251	359	-	Windows

Tabelle 46. Brasilien, Gebietskennung: BR

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
819	S-1	ISO8859-1	55	pt_BR	AIX
850	S-1	IBM-850	55	-	AIX
923	S-1	ISO8859-15	55	pt_BR.8859-15	AIX
1208	N-1	UTF-8	55	PT_BR	AIX
37	S-1	IBM-37	55	-	Host
1140	S-1	IBM-1140	55	-	Host
819	S-1	ISO8859-1	55	-	HP-UX
923	S-1	ISO8859-15	55	-	HP-UX
819	S-1	ISO-8859-1	55	pt_BR	Linux
923	S-1	ISO-8859-15	55	-	Linux
850	S-1	IBM-850	55	-	OS/2
819	S-1	ISO8859-1	55	pt_BR	SCO
819	S-1	ISO8859-1	55	pt_BR	Solaris
923	S-1	ISO8859-15	55	-	Solaris
1252	S-1	1252	55	-	Windows

Tabelle 47. Kanada, Gebietskennung: CA

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
819	S-1	ISO8859-1	1	fr_CA	AIX
850	S-1	IBM-850	1	Fr_CA	AIX
923	S-1	ISO8859-15	1	fr_CA.8859-15	AIX
1208	N-1	UTF-8	1	FR_CA	AIX
37	S-1	IBM-37	1	-	Host
1140	S-1	IBM-1140	1	-	Host
819	S-1	iso88591	1	fr_CA.iso88591	HP-UX
923	S-1	iso885915	1	-	HP-UX
1051	S-1	roman8	1	fr_CA.roman8	HP-UX
819	S-1	ISO-8859-1	1	en_CA	Linux
923	S-1	ISO-8859-15	1	-	Linux
850	S-1	IBM-850	1	-	OS/2
819	S-1	ISO8859-1	1	en_CA	SCO
819	S-1	ISO8859-1	1	fr_CA	SCO
819	S-1	ISO8859-1	1	en_CA	Solaris
923	S-1	ISO8859-15	1	-	Solaris
1252	S-1	1252	1	-	Windows

Tabelle 48. Kanada (Französisch), Gebietskennung: CA

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
863	S-1	IBM-863	2	-	OS/2

Tabelle 49. China (VRC), Gebietskennung: CN

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
1383	D-4	IBM-eucCN	86	zh_CN	AIX
1386	D-4	GBK	86	Zh_CN.GBK	AIX
1208	N-1	UTF-8	86	ZH_CN	AIX
935	D-4	IBM-935	86	-	Host
1388	D-4	IBM-1388	86	-	Host
1383	D-4	hp15CN	86	zh_CN.hp15CN	HP-UX
1383	D-4	GBK	86	zh_CN.GBK	Linux
1381	D-4	IBM-1381	86	-	OS/2
1386	D-4	GBK	86	-	OS/2
1383	D-4	eucCN	86	zh_CN	SCO
1383	D-4	eucCN	86	zh_CN.eucCN	SCO
1383	D-4	gb2312	86	zh	Solaris
1208	N-1	UTF-8	86	zh.UTF-8	Solaris
1381	D-4	IBM-1381	86	-	Windows
1386	D-4	GBK	86	-	Windows
1392/5488	D-4		86	-	

Siehe Anmerkung 1 auf Seite 284.

Tabelle 50. Kroatien, Gebietskennung: HR

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
912	S-2	ISO8859-2	385	hr_HR	AIX
1208	N-1	UTF-8	385	HR_HR	AIX
870	S-2	IBM-870	385	-	Host
1153	S-2	IBM-1153	385	-	Host
912	S-2	iso88592	385	hr_HR.iso88592	HP-UX
912	S-2	ISO-8859-2	385	hr_HR	Linux
852	S-2	IBM-852	385	-	OS/2
912	S-2	ISO8859-2	385	hr_HR.ISO8859-2	SCO
1250	S-2	1250	385	-	Windows

Tabelle 51. Tschechische Republik, Gebietskennung: CZ

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
912	S-2	ISO8859-2	421	cs_CZ	AIX
1208	N-1	UTF-8	421	CS_CZ	AIX
870	S-2	IBM-870	421	-	Host
1153	S-2	IBM-1153	421	-	Host
912	S-2	iso88592	421	cs_CZ.iso88592	HP-UX
912	S-2	ISO-8859-2	421	cs_CZ	Linux
852	S-2	IBM-852	421	-	OS/2
912	S-2	ISO8859-2	421	cs_CZ.ISO8859-2	SCO
1250	S-2	1250	421	-	Windows

Tabelle 52. Dänemark, Gebietskennung: DK

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
819	S-1	ISO8859-1	45	da_DK	AIX
850	S-1	IBM-850	45	Da_DK	AIX
923	S-1	ISO8859-15	45	da_DK.8859-15	AIX
1208	N-1	UTF-8	45	DA_DK	AIX
277	S-1	IBM-277	45	-	Host
1142	S-1	IBM-1142	45	-	Host
819	S-1	iso88591	45	da_DK.iso88591	HP-UX
923	S-1	iso885915	45	_	HP-UX
1051	S-1	roman8	45	da_DK.roman8	HP-UX
819	S-1	ISO-8859-1	45	da_DK	Linux
923	S-1	ISO-8859-15	45	-	Linux
850	S-1	IBM-850	45	-	OS/2
819	S-1	ISO8859-1	45	da	SCO
819	S-1	ISO8859-1	45	da_DA	SCO
819	S-1	ISO8859-1	45	da_DK	SCO
819	S-1	ISO8859-1	45	da	Solaris
923	S-1	ISO8859-15	45	da.ISO8859-15	Solaris
1252	S-1	1252	45	-	Windows

Tabelle 53. Estland, Gebietskennung: EE

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
922	S-10	IBM-922	372	Et_EE	AIX

Tabelle 53. Estland, Gebietskennung: EE (Forts.)

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
1208	N-1	UTF-8	372	ET_EE	AIX
1122	S-10	IBM-1122	372	-	Host
1157	S-10	IBM-1157	372	-	Host
922	S-10	IBM-922	372	-	OS/2
1257	S-10	1257	372	-	Windows

Tabelle 54. Finnland, Gebietskennung: FI

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
819	S-1	ISO8859-1	358	fi_FI	AIX
850	S-1	IBM-850	358	Fi_FI	AIX
923	S-1	ISO8859-15	358	fi_FI.8859-15	AIX
1208	N-1	UTF-8	358	FI_FI	AIX
278	S-1	IBM-278	358	-	Host
1143	S-1	IBM-1143	358	-	Host
819	S-1	iso88591	358	fi_FI.iso88591	HP-UX
923	S-1	iso885915	358	-	HP-UX
1051	S-1	roman8	358	fi-FI.roman8	HP-UX
819	S-1	ISO-8859-1	358	fi_FI	Linux
923	S-1	ISO-8859-15	358	fi_FI@euro	Linux
437	S-1	IBM-437	358	-	OS/2
850	S-1	IBM-850	358	-	OS/2
819	S-1	ISO8859-1	358		SCO
819	S-1	ISO8859-1	358	fi_FI	SCO
819	S-1	ISO8859-1	358	sv_FI	SCO
819	S-1	ISO8859-1	358	-	Solaris
923	S-1	ISO8859-15	358	fi.ISO8859-15	Solaris
1252	S-1	1252	358	-	Windows

Tabelle 55. Mazedonien (ehm. jugosl. Rep.), Gebietskennung: MK

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
915	S-5	ISO8859-5	389	mk_MK	AIX
1208	N-1	UTF-8	389	MK_MK	AIX
1025	S-5	IBM-1025	389	-	Host
1154	S-5	IBM-1154	389	-	Host
915	S-5	iso88595	389	-	HP-UX
855	S-5	IBM-855	389	-	OS/2
915	S-5	ISO8859-5	389	-	OS/2
1251	S-5	1251	389	-	Windows

Tabelle 56. Frankreich, Gebietskennung: FR

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
819	S-1	ISO8859-1	33	fr_FR	AIX
850	S-1	IBM-850	33	Fr_FR	AIX
923	S-1	ISO8859-15	33	fr_FR.8859-15	AIX
1208	N-1	UTF-8	33	FR_FR	AIX
297	S-1	IBM-297	33	-	Host

Tabelle 56. Frankreich, Gebietskennung: FR (Forts.)

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
1147	S-1	IBM-1147	33	-	Host
819	S-1	iso88591	33	fr_FR.iso88591	HP-UX
923	S-1	iso885915	33	-	HP-UX
1051	S-1	roman8	33	fr_FR.roman8	HP-UX
819	S-1	ISO-8859-1	33	fr_FR	Linux
923	S-1	ISO-8859-15	33	fr_FR@euro	Linux
437	S-1	IBM-437	33	-	OS/2
850	S-1	IBM-850	33	-	OS/2
819	S-1	ISO8859-1	33		SCO
819	S-1	ISO8859-1	33	fr_FR	SCO
819	S-1	ISO8859-1	33		Solaris
923	S-1	ISO8859-15	33	fr.ISO8859-15	Solaris
1208	N-1	UTF-8	33	fr.UTF-8	Solaris
1252	S-1	1252	33	-	Windows

Tabelle 57. Deutschland, Gebietskennung: DE

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
819	S-1	ISO8859-1	49	de_DE	AIX
850	S-1	IBM-850	49	De_DE	AIX
923	S-1	ISO8859-15	49	de_DE.8859-15	AIX
1208	N-1	UTF-8	49	DE_DE	AIX
273	S-1	IBM-273	49	-	Host
1141	S-1	IBM-1141	49	-	Host
819	S-1	iso88591	49	de_DE.iso88591	HP-UX
923	S-1	iso885915	49	_	HP-UX
1051	S-1	roman8	49	de_DE.roman8	HP-UX
819	S-1	ISO-8859-1	49	de_DE	Linux
923	S-1	ISO-8859-15	49	de_DE@euro	Linux
437	S-1	IBM-437	49	-	OS/2
850	S-1	IBM-850	49	-	OS/2
819	S-1	ISO8859-1	49		SCO
819	S-1	ISO8859-1	49	de_DE	SCO
819	S-1	ISO8859-1	49		Solaris
923	S-1	ISO8859-15	49	de.ISO8859-15	Solaris
1208	N-1	UTF-8	49	de.UTF-8	Solaris
1252	S-1	1252	49	-	Windows

Tabelle 58. Griechenland, Gebietskennung: GR

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
813	S-7	ISO8859-7	30	el_GR	AIX
1208	N-1	UTF-8	30	EL_GR	AIX
423	S-7	IBM-423	30	-	Host
875	S-7	IBM-875	30	-	Host
813	S-7	iso88597	30	el_GR.iso88597	HP-UX
813	S-7	ISO-8859-7	30	el_GR	Linux
813	S-7	ISO8859-7	30	-	OS/2
869	S-7	IBM-869	30	-	OS/2
813	S-7	ISO8859-7	30	el_GR.ISO8859-7	SCO

Tabelle 58. Griechenland, Gebietskennung: GR (Forts.)

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
737	S-7	737	30	-	Windows
1253	S-7	1253	30	-	Windows

Tabelle 59. Ungarn, Gebietskennung: HU

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
912	S-2	ISO8859-2	36	hu_HU	AIX
1208	N-1	UTF-8	36	HU_HU	AIX
870	S-2	IBM-870	36	-	Host
1153	S-2	IBM-1153	36	-	Host
912	S-2	iso88592	36	hu_HU.iso88592	HP-UX
912	S-2	ISO-8859-2	36	hu_HU	Linux
852	S-2	IBM-852	36	-	OS/2
912	S-2	ISO8859-2	36	hu_HU.ISO8859-2	SCO
1250	S-2	1250	36	-	Windows

Tabelle 60. Island, Gebietskennung: IS

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
819	S-1	ISO8859-1	354	is_IS	AIX
850	S-1	IBM-850	354	Is_IS	AIX
923	S-1	ISO8859-15	354	is_IS.8859-15	AIX
1208	N-1	UTF-8	354	IS_IS	AIX
871	S-1	IBM-871	354	-	Host
1149	S-1	IBM-1149	354	-	Host
819	S-1	iso88591	354	is_IS.iso88591	HP-UX
923	S-1	iso885915	354	-	HP-UX
1051	S-1	roman8	354	is_IS.roman8	HP-UX
819	S-1	ISO-8859-1	354	is_IS	Linux
923	S-1	ISO-8859-15	354	-	Linux
850	S-1	IBM-850	354	-	OS/2
819	S-1	ISO8859-1	354		SCO
819	S-1	ISO8859-1	354	is_IS	SCO
819	S-1	ISO8859-1	354	-	Solaris
923	S-1	ISO8859-15	354	-	Solaris
1252	S-1	1252	354	-	Windows

Tabelle 61. Indien, Gebietskennung: IN

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
806	S-13	IBM-806	91	hi_IN	-
1137	S-13	IBM-1137	91	-	Host

Tabelle 62. Indonesien, Gebietskennung: ID

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
1252	S-1	1252	62	-	Windows

Tabelle 63. Irland, Gebietskennung: IE

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
819	S-1	ISO8859-1	353	-	AIX
850	S-1	IBM-850	353	-	AIX
923	S-1	ISO8859-15	353	-	AIX
1208	N-1	UTF-8	353	-	AIX
285	S-1	IBM-285	353	-	Host
1146	S-1	IBM-1146	353	-	Host
819	S-1	iso88591	353	-	HP-UX
923	S-1	iso885915	353	-	HP-UX
1051	S-1	roman8	353	-	HP-UX
819	S-1	ISO-8859-1	353	en_IE	Linux
923	S-1	ISO-8859-15	353	en_IE@euro	Linux
437	S-1	IBM-437	353	-	OS/2
850	S-1	IBM-850	353	-	OS/2
819	S-1	ISO8859-1	353	en_IE.ISO8859-1	SCO
819	S-1	ISO8859-1	353	en_IE	Solaris
923	S-1	ISO8859-15	353	en_IE.ISO8859-15	Solaris
1252	S-1	1252	353	-	Windows

Tabelle 64. Israel, Gebietskennung: IL

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
856	S-8	IBM-856	972	Iw_IL	AIX
916	S-8	ISO8859-8	972	iw_IL	AIX
1208	N-1	UTF-8	972	HE-IL	AIX
916	S-8	ISO-8859-8	972	iw_IL	Linux
424	S-8	IBM-424	972	-	Host
862	S-8	IBM-862	972	-	OS/2
1255	S-8	1255	972	-	Windows

Tabelle 65. Italien, Gebietskennung: IT

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
819	S-1	ISO8859-1	39	it_IT	AIX
850	S-1	IBM-850	39	It_IT	AIX
923	S-1	ISO8859-15	39	it_IT.8859-15	AIX
1208	N-1	UTF-8	39	It_IT	AIX
280	S-1	IBM-280	39	-	Host
1144	S-1	IBM-1144	39	-	Host
819	S-1	iso88591	39	it_IT.iso88591	HP-UX
923	S-1	iso885915	39	_	HP-UX
1051	S-1	roman8	39	it_IT.roman8	HP-UX
819	S-1	ISO-8859-1	39	it_IT	Linux
923	S-1	ISO-8859-15	39	it_IT@euro	Linux
437	S-1	IBM-437	39	-	OS/2
850	S-1	IBM-850	39	-	OS/2
819	S-1	ISO8859-1	39		SCO
819	S-1	ISO8859-1	39	it_IT	SCO
819	S-1	ISO8859-1	39		Solaris
923	S-1	ISO8859-15	39	it.ISO8859-15	Solaris
1208	N-1	UTF-8	39	it.UTF-8	Solaris

Tabelle 65. Italien, Gebietskennung: IT (Forts.)

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
1252	S-1	1252	39	-	Windows

Tabelle 66. Japan, Gebietskennung: JP

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
932	D-1	IBM-932	81	Ja_JP	AIX
943	D-1	IBM-943	81	Ja_JP	AIX
Siehe Anmerkung 2 auf Seite 284.					
954	D-1	IBM-eucJP	81	ja_JP	AIX
1208	N-1	UTF-8	81	JA_JP	AIX
930	D-1	IBM-930	81	-	Host
939	D-1	IBM-939	81	-	Host
5026	D-1	IBM-5026	81	-	Host
5035	D-1	IBM-5035	81	-	Host
1390	D-1		81	-	Host
1399	D-1		81	-	Host
954	D-1	eucJP	81	ja_JP.eucJP	HP-UX
5039	D-1	SJIS	81	ja_JP.SJIS	HP-UX
954	D-1	EUC-JP	81	ja_JP	Linux
932	D-1	IBM-932	81	-	OS/2
942	D-1	IBM-942	81	-	OS/2
943	D-1	IBM-943	81	-	OS/2
954	D-1	eucJP	81	ja	SCO
954	D-1	eucJP	81	ja_JP	SCO
954	D-1	eucJP	81	ja_JP.EUC	SCO
954	D-1	eucJP	81	ja_JP.eucJP	SCO
943	D-1	IBM-943	81	ja_JP.PCK	Solaris
954	D-1	eucJP	81	ja	Solaris
954	D-1	eucJP	81	Japanisch	Solaris
1208	N-1	UTF-8	81	ja_JP.UTF-8	Solaris
943	D-1	IBM-943	81	-	Windows
1394	D-1		81	-	
Siehe Anmerkung 3 auf Seite 284.					

Tabelle 67. Kasachstan, Gebietskennung: KZ

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
1251	S-5	1251	7	-	Windows

Tabelle 68. Korea, Süd, Gebietskennung: KR

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
970	D-3	IBM-eucKR	82	ko_KR	AIX
1208	N-1	UTF-8	82	KO_KR	AIX
933	D-3	IBM-933	82	-	Host
1364	D-3	IBM-1364	82	-	Host
970	D-3	eucKR	82	ko_KR.eucKR	HP-UX
970	D-3	EUC-KR	82	ko_KR	Linux
949	D-3	IBM-949	82	-	OS/2

Tabelle 68. Korea, Süd, Gebietskennung: KR (Forts.)

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
970	D-3	eucKR	82	ko_KR.eucKR	SGI
970	D-3	5601	82	ko	Solaris
1208	N-1	UTF-8	82	ko.UTF-8	Solaris
1363	D-3	1363	82	-	Windows

Tabelle 69. Lateinamerika, Gebietskennung: Lat

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
819	S-1	ISO8859-1	3	-	AIX
850	S-1	IBM-850	3	-	AIX
923	S-1	ISO8859-15	3	-	AIX
1208	N-1	UTF-8	3	-	AIX
284	S-1	IBM-284	3	-	Host
1145	S-1	IBM-1145	3	-	Host
819	S-1	iso88591	3	-	HP-UX
923	S-1	iso885915	3	-	HP-UX
1051	S-1	roman8	3	-	HP-UX
819	S-1	ISO-8859-1	3	-	Linux
923	S-1	ISO-8859-15	3	-	Linux
437	S-1	IBM-437	3	-	OS/2
850	S-1	IBM-850	3	-	OS/2
819	S-1	ISO8859-1	3	-	Solaris
923	S-1	ISO8859-15	3	-	Solaris
1252	S-1	1252	3	-	Windows

Tabelle 70. Lettland, Gebietskennung: LV

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
921	S-10	IBM-921	371	Lv_LV	AIX
1208	N-1	UTF-8	371	LV_LV	AIX
1112	S-10	IBM-1112	371	-	Host
1156	S-10	IBM-1156	371	-	Host
921	S-10	IBM-921	371	-	OS/2
1257	S-10	1257	371	-	Windows

Tabelle 71. Litauen, Gebietskennung: LT

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
921	S-10	IBM-921	370	Lt_LT	AIX
1208	N-1	UTF-8	370	LT_LT	AIX
1112	S-10	IBM-1112	370	-	Host
1156	S-10	IBM-1156	370	-	Host
921	S-10	IBM-921	370	-	OS/2
1257	S-10	1257	370	-	Windows

Tabelle 72. Malaysia, Gebietskennung: ID

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
1252	S-1	1252	60	-	Windows

Tabelle 73. Niederlande, Gebietskennung: NL

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
819	S-1	ISO8859-1	31	nl_NL	AIX
850	S-1	IBM-850	31	Nl_NL	AIX
923	S-1	ISO8859-15	31	nl_NL.8859-15	AIX
1208	N-1	UTF-8	31	NL_NL	AIX
37	S-1	IBM-37	31	-	Host
1140	S-1	IBM-1140	31	-	Host
819	S-1	iso88591	31	nl_NL.iso88591	HP-UX
923	S-1	iso885915	31	_	HP-UX
1051	S-1	roman8	31	nl_NL.roman8	HP-UX
819	S-1	ISO-8859-1	31	nl_NL	Linux
923	S-1	ISO-8859-15	31	nl_NL@euro	Linux
437	S-1	IBM-437	31	-	OS/2
850	S-1	IBM-850	31	-	OS/2
819	S-1	ISO8859-1	31	nl	SCO
819	S-1	ISO8859-1	31	nl_NL	SCO
819	S-1	ISO8859-1	31	nl	Solaris
923	S-1	ISO8859-15	31	nl.ISO8859-15	Solaris
1252	S-1	1252	31	-	Windows

Tabelle 74. Neuseeland, Gebietskennung: NZ

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
819	S-1	ISO8859-1	64	-	AIX
850	S-1	IBM-850	64	-	AIX
923	S-1	ISO8859-15	64	-	AIX
1208	N-1	UTF-8	64	-	AIX
37	S-1	IBM-37	64	-	Host
1140	S-1	IBM-1140	64	-	Host
819	S-1	ISO8859-1	64	-	HP-UX
923	S-1	ISO8859-15	64	-	HP-UX
850	S-1	IBM-850	64	-	OS/2
819	S-1	ISO8859-1	64	en_NZ	SCO
819	S-1	ISO8859-1	64	en_NZ	Solaris
923	S-1	ISO8859-15	64	-	Solaris
1252	S-1	1252	64	-	Windows

Tabelle 75. Norwegen, Gebietskennung: NO

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
819	S-1	ISO8859-1	47	no_NO	AIX
850	S-1	IBM-850	47	No_NO	AIX
923	S-1	ISO8859-15	47	no_NO.8859-15	AIX
1208	N-1	UTF-8	47	NO_NO	AIX
277	S-1	IBM-277	47	-	Host

Tabelle 75. Norwegen, Gebietskennung: NO (Forts.)

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
1142	S-1	IBM-1142	47	-	Host
819	S-1	iso88591	47	no_NO.iso88591	HP-UX
923	S-1	iso885915	47	-	HP-UX
1051	S-1	roman8	47	no_NO.roman8	HP-UX
819	S-1	ISO-8859-1	47	no_NO	Linux
923	S-1	ISO-8859-15	47	-	Linux
850	S-1	IBM-850	47	-	OS/2
819	S-1	ISO8859-1	47	no	SCO
819	S-1	ISO8859-1	47	no_NO	SCO
819	S-1	ISO8859-1	47	no	Solaris
923	S-1	ISO8859-15	47	-	Solaris
1252	S-1	1252	47	-	Windows

Tabelle 76. Polen, Gebietskennung: PL

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
912	S-2	ISO8859-2	48	pl_PL	AIX
1208	N-1	UTF-8	48	PL_PL	AIX
870	S-2	IBM-870	48	-	Host
1153	S-2	IBM-1153	48	-	Host
912	S-2	iso88592	48	pl_PL.iso88592	HP-UX
912	S-2	ISO-8859-2	48	pl_PL	Linux
852	S-2	IBM-852	48	-	OS/2
912	S-2	ISO8859-2	48	pl_PL.ISO8859-2	SCO
1250	S-2	1250	48	-	Windows

Tabelle 77. Portugal, Gebietskennung: PT

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
819	S-1	ISO8859-1	351	pt_PT	AIX
850	S-1	IBM-850	351	Pt_PT	AIX
923	S-1	ISO8859-15	351	pt_PT.8859-15	AIX
1208	N-1	UTF-8	351	PT_PT	AIX
37	S-1	IBM-37	351	-	Host
1140	S-1	IBM-1140	351	-	Host
819	S-1	iso88591	351	pt_PT.iso88591	HP-UX
923	S-1	iso885915	351	-	HP-UX
1051	S-1	roman8	351	pt_PT.roman8	HP-UX
819	S-1	ISO-8859-1	351	pt_PT	Linux
923	S-1	ISO-8859-15	351	pt_PT@euro	Linux
850	S-1	IBM-850	351	-	OS/2
860	S-1	IBM-860	351	-	OS/2
819	S-1	ISO8859-1	351	pt	SCO
819	S-1	ISO8859-1	351	pt_PT	SCO
819	S-1	ISO8859-1	351	pt	Solaris
923	S-1	ISO8859-15	351	pt.ISO8859-15	Solaris
1252	S-1	1252	351	-	Windows

Tabelle 78. Rumänien, Gebietskennung: RO

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
912	S-2	ISO8859-2	40	ro_RO	AIX
1208	N-1	UTF-8	40	RO_RO	AIX
870	S-2	IBM-870	40	-	Host
1153	S-2	IBM-1153	40	-	Host
912	S-2	iso88592	40	ro_RO.iso88592	HP-UX
912	S-2	ISO-8859-2	40	ro_RO	Linux
852	S-2	IBM-852	40	-	OS/2
912	S-2	ISO8859-2	40	ro_RO.ISO8859-2	SCO
1250	S-2	1250	40	-	Windows

Tabelle 79. Russland, Gebietskennung: RU

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
915	S-5	ISO8859-5	7	ru_RU	AIX
1208	N-1	UTF-8	7	RU_RU	AIX
1025	S-5	IBM-1025	7	-	Host
1154	S-5	IBM-1154	7	-	Host
915	S-5	iso88595	7	ru_RU.iso88595	HP-UX
878	S-5	KOI8-R	7	ru_RU.koi8-r	Linux, Solaris
915	S-5	ISO-8859-5	7	ru_RU	Linux
866	S-5	IBM-866	7	-	OS/2
915	S-5	ISO8859-5	7	-	OS/2
915	S-5	ISO8859-5	7	ru_RU.ISO8859-5	SCO
1251	S-5	1251	7	-	Windows

Tabelle 80. Serbien/Montenegro, Gebietskennung: SP

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
915	S-5	ISO8859-5	381	sr_SP	AIX
1208	N-1	UTF-8	381	SR_SP	AIX
1025	S-5	IBM-1025	381	-	Host
1154	S-5	IBM-1154	381	-	Host
915	S-5	iso88595	381	-	HP-UX
855	S-5	IBM-855	381	-	OS/2
915	S-5	ISO8859-5	381	-	OS/2
1251	S-5	1251	381	-	Windows

Tabelle 81. Slowakei, Gebietskennung: SK

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
912	S-2	ISO8859-2	422	sk_SK	AIX
1208	N-1	UTF-8	422	SK_SK	AIX
870	S-2	IBM-870	422	-	Host
1153	S-2	IBM-1153	422	-	Host
912	S-2	iso88592	422	sk_SK.iso88592	HP-UX
852	S-2	IBM-852	422	-	OS/2
912	S-2	ISO8859-2	422	sk_SK.ISO8859-2	SCO
1250	S-2	1250	422	-	Windows

Tabelle 82. Slowenien, Gebietskennung: SI

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
912	S-2	ISO8859-2	386	sl_SI	AIX
1208	N-1	UTF-8	386	SL_SI	AIX
870	S-2	IBM-870	386	-	Host
1153	S-2	IBM-1153	386	-	Host
912	S-2	iso88592	386	sl_SI.iso88592	HP-UX
912	S-2	ISO-8859-2	386	sl_SI	Linux
852	S-2	IBM-852	386	-	OS/2
912	S-2	ISO8859-2	386	sl_SI.ISO8859-2	SCO
1250	S-2	1250	386	-	Windows

Tabelle 83. Südafrika, Gebietskennung: ZA

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
819	S-1	ISO8859-1	27	en_ZA	AIX
850	S-1	IBM-850	27	En_ZA	AIX
923	S-1	ISO8859-15	27	en_ZA.8859-15	AIX
1208	N-1	UTF-8	27	EN_ZA	AIX
285	S-1	IBM-285	27	-	Host
1146	S-1	IBM-1146	27	-	Host
819	S-1	iso88591	27	-	HP-UX
923	S-1	iso885915	27	-	HP-UX
1051	S-1	roman8	27	-	HP-UX
437	S-1	IBM-437	27	-	OS/2
850	S-1	IBM-850	27	-	OS/2
819	S-1	ISO8859-1	27	en_ZA.ISO8859-1	SCO
819	S-1	ISO8859-1	27	-	Solaris
923	S-1	ISO8859-15	27	-	Solaris
1252	S-1	1252	27	-	Windows

Tabelle 84. Spanien, Gebietskennung: ES

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
819	S-1	ISO8859-1	34	es_ES	AIX
850	S-1	IBM-850	34	Es_ES	AIX
923	S-1	ISO8859-15	34	es_ES.8859-15	AIX
1208	N-1	UTF-8	34	ES_ES	AIX
284	S-1	IBM-284	34	-	Host
1145	S-1	IBM-1145	34	-	Host
819	S-1	iso88591	34	es_ES.iso88591	HP-UX
923	S-1	iso885915	34	-	HP-UX
1051	S-1	roman8	34	es_ES.roman8	HP-UX
819	S-1	ISO-8859-1	34	es_ES	Linux
923	S-1	ISO-8859-15	34	es_ES@euro	Linux
437	S-1	IBM-437	34	-	OS/2
850	S-1	IBM-850	34	-	OS/2
819	S-1	ISO8859-1	34	es	SCO
819	S-1	ISO8859-1	34	es_ES	SCO
819	S-1	ISO8859-1	34	es	Solaris

Tabelle 84. Spanien, Gebietskennung: ES (Forts.)

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
923	S-1	ISO8859-15	34	es.ISO8859-15	Solaris
1208	N-1	UTF-8	34	es.UTF-8	Solaris
1252	S-1	1252	34	-	Windows

Tabelle 85. Spanien (Katalanisch), Gebietskennung: ES

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
819	S-1	ISO8859-1	34	ca_ES	AIX
850	S-1	IBM-850	34	Ca_ES	AIX
923	S-1	ISO8859-15	34	ca_ES.8859-15	AIX
1208	N-1	UTF-8	34	CA_ES	AIX

Tabelle 86. Schweden, Gebietskennung: SE

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
819	S-1	ISO8859-1	46	sv_SE	AIX
850	S-1	IBM-850	46	Sv_SE	AIX
923	S-1	ISO8859-15	46	sv_SE.8859-15	AIX
1208	N-1	UTF-8	46	SV_SE	AIX
278	S-1	IBM-278	46	-	Host
1143	S-1	IBM-1143	46	-	Host
819	S-1	iso88591	46	sv_SE.iso88591	HP-UX
923	S-1	iso885915	46	-	HP-UX
1051	S-1	roman8	46	sv_SE.roman8	HP-UX
819	S-1	ISO-8859-1	46	sv_SE	Linux
923	S-1	ISO-8859-15	46	-	Linux
437	S-1	IBM-437	46	-	OS/2
850	S-1	IBM-850	46	-	OS/2
819	S-1	ISO8859-1	46	sv	SCO
819	S-1	ISO8859-1	46	sv_SE	SCO
819	S-1	ISO8859-1	46	sv	Solaris
923	S-1	ISO8859-15	46	sv.ISO8859-15	Solaris
1208	N-1	UTF-8	46	sv.UTF-8	Solaris
1252	S-1	1252	46	-	Windows

Tabelle 87. Schweiz, Gebietskennung: CH

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
819	S-1	ISO8859-1	41	de_CH	AIX
850	S-1	IBM-850	41	De_CH	AIX
923	S-1	ISO8859-15	41	de_CH.8859-15	AIX
1208	N-1	UTF-8	41	DE_CH	AIX
500	S-1	IBM-500	41	-	Host
1148	S-1	IBM-1148	41	-	Host
819	S-1	iso88591	41	-	HP-UX
923	S-1	iso885915	41	-	HP-UX
1051	S-1	roman8	41	-	HP-UX
819	S-1	ISO-8859-1	41	de_CH	Linux
923	S-1	ISO-8859-15	41	-	Linux

Tabelle 87. Schweiz, Gebietskennung: CH (Forts.)

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
437	S-1	IBM-437	41	-	OS/2
850	S-1	IBM-850	41	-	OS/2
819	S-1	ISO8859-1	41	de_CH	SCO
819	S-1	ISO8859-1	41	fr_CH	SCO
819	S-1	ISO8859-1	41	it_CH	SCO
819	S-1	ISO8859-1	41	de_CH	Solaris
923	S-1	ISO8859-15	41	-	Solaris
1252	S-1	1252	41	-	Windows

Tabelle 88. Taiwan, Gebietskennung: TW

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
950	D-2	big5	88	Zh_TW	AIX
Siehe Anmerkung 8 auf Seite 285.					
964	D-2	IBM-eucTW	88	zh_TW	AIX
1208	N-1	UTF-8	88	ZH_TW	AIX
937	D-2	IBM-937	88	-	Host
1371	D-2	IBM-1371	88	-	Host
950	D-2	big5	88	zh_TW.big5	HP-UX
964	D-2	eucTW	88	zh_TW.eucTW	HP-UX
950	D-2	BIG5	88	zh_TW	Linux
938	D-2	IBM-938	88	-	OS/2
948	D-2	IBM-948	88	-	OS/2
950	D-2	big5	88	-	OS/2
950	D-2	big5	88	zh_TW.BIG5	Solaris
964	D-2	cns11643	88	zh_TW	Solaris
1208	N-1	UTF-8	88	zh_TW.UTF-8	Solaris
950	D-2	big5	88	-	Windows
Siehe Anmerkung 8 auf Seite 285.					

Tabelle 89. Thailand, Gebietskennung: TH

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
874	S-20	TIS620-1	66	th_TH	AIX
1208	N-1	UTF-8	66	TH_TH	AIX
838	S-20	IBM-838	66	-	Host
1160	S-20	IBM-1160	66	-	Host
874	S-20	tis620	66	th_TH.tis620	HP-UX
874	S-20	TIS620-1	66	-	OS/2
874	S-20	TIS620-1	66	-	Windows

Tabelle 90. Türkei, Gebietskennung: TR

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
920	S-9	ISO8859-9	90	tr_TR	AIX
1208	N-1	UTF-8	90	TR_TR	AIX
1026	S-9	IBM-1026	90	-	Host
1155	S-9	IBM-1155	90	-	Host
920	S-9	iso88599	90	tr_TR.iso88599	HP-UX

Tabelle 90. Türkei, Gebietskennung: TR (Forts.)

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
920	S-9	ISO-8859-9	90	tr_TR	Linux
857	S-9	IBM-857	90	-	OS/2
920	S-9	ISO8859-9	90	tr_TR.ISO8859-9	SCO
1254	S-9	1254	90	-	Windows

Tabelle 91. Großbritannien, Gebietskennung: GB

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
819	S-1	ISO8859-1	44	en_GB	AIX
850	S-1	IBM-850	44	En_GB	AIX
923	S-1	ISO8859-15	44	en_GB.8859-15	AIX
1208	N-1	UTF-8	44	EN_GB	AIX
285	S-1	IBM-285	44	-	Host
1146	S-1	IBM-1146	44	-	Host
819	S-1	iso88591	44	en_GB.iso88591	HP-UX
923	S-1	iso885915	44	-	HP-UX
1051	S-1	roman8	44	en_GB.roman8	HP-UX
819	S-1	ISO-8859-1	44	en_GB	Linux
923	S-1	ISO-8859-15	44	-	Linux
437	S-1	IBM-437	44	-	OS/2
850	S-1	IBM-850	44	-	OS/2
819	S-1	ISO8859-1	44	en_GB	SCO
819	S-1	ISO8859-1	44	en	SCO
819	S-1	ISO8859-1	44	en_GB	Solaris
923	S-1	ISO8859-15	44	en_GB.ISO8859-15	Solaris
1252	S-1	1252	44	-	Windows

Tabelle 92. Ukraine, Gebietskennung: UA

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
1124	S-12	IBM-1124	380	Uk_UA	AIX
1208	N-1	UTF-8	380	UK_UA	AIX
1123	S-12	IBM-1123	380	-	Host
1158	S-12	IBM-1158	380	-	Host
1168	S-12	KOI8-U	380	uk_UA.koi8u	Linux
1125	S-12	IBM-1125	380	-	OS/2
1251	S-12	1251	380	-	Windows

Tabelle 93. Vereinigte Staaten von Amerika, Gebietskennung: US

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
819	S-1	ISO8859-1	1	en_US	AIX
850	S-1	IBM-850	1	En_US	AIX
923	S-1	ISO8859-15	1	en_US.8859-15	AIX
1208	N-1	UTF-8	1	EN_US	AIX
37	S-1	IBM-37	1	-	Host
1140	S-1	IBM-1140	1	-	Host
819	S-1	iso88591	1	en_US.iso88591	HP-UX
923	S-1	iso885915	1	-	HP-UX

Tabelle 93. Vereinigte Staaten von Amerika, Gebietskennung: US (Forts.)

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
1051	S-1	roman8	1	en_US.roman8	HP-UX
819	S-1	ISO-8859-1	1	en_US	Linux
923	S-1	ISO-8859-15	1	-	Linux
437	S-1	IBM-437	1	-	OS/2
850	S-1	IBM-850	1	-	OS/2
819	S-1	ISO8859-1	1	en_US	SCO
819	S-1	ISO8859-1	1	en_US	SGI
819	S-1	ISO8859-1	1	en_US	Solaris
923	S-1	ISO8859-15	1	en_US.ISO8859-15	Solaris
1208	N-1	UTF-8	1	en_US.UTF-8	Solaris
1252	S-1	1252	1	-	Windows

Tabelle 94. Vietnam, Gebietskennung: VN

Codepage	Gruppe	Codierter Zeichens.	Gebiets-code	Länder-einstellung	Betriebs-system
1129	S-11	IBM-1129	84	Vi_VN	AIX
1208	N-1	UTF-8	84	VI_VN	AIX
1130	S-11	IBM-1130	84	-	Host
1164	S-11	IBM-1164	84	-	Host
1129	S-11	IBM-1129	84	-	OS/2
1258	S-11	1258	84	-	Windows

Anmerkungen:

1. Die CCSIDs 1392 und 5488 (GB 18030) können nur mit den Dienstprogrammen LOAD und IMPORT zum Versetzen von Daten aus den CCSIDs 1392 und 5488 in eine DB2 UDB-Unicode-Datenbank bzw. mit dem Dienstprogramm EXPORT zum Exportieren aus einer DB2 UDB-Unicode-Datenbank in die CCSIDs 1392 oder 5488 verwendet werden.
2. Unter AIX 4.3 oder einer späteren Version ist die Codepage 943. Bei Verwendung von AIX 4.2 oder einer früheren Version ist die Codepage 932.
3. Codepage 1394 (Shift JIS X0213) kann nur mit den Dienstprogrammen LOAD oder IMPORT zum Versetzen von Daten aus Codepage 1394 in eine DB2 UDB-Unicode-Datenbank bzw. mit dem Dienstprogramm EXPORT zum Exportieren aus einer DB2 UDB-Unicode-Datenbank in Codepage 1394 verwendet werden.
4. Übersicht der arabischen Länder/Regionen (AA):
   - Arabisch (Saudi-Arabien)
   - Arabisch (Irak)
   - Arabisch (Ägypten)
   - Arabisch (Libyen)
   - Arabisch (Algerien)
   - Arabisch (Marokko)
   - Arabisch (Tunesien)
   - Arabisch (Oman)
   - Arabisch (Jemen)
   - Arabisch (Syrien)
   - Arabisch (Jordanien)
   - Arabisch (Libanon)

- Arabisch (Kuwait)
- Arabisch (Vereinigte Arabische Emirate)
- Arabisch (Bahrain)
- Arabisch (Katar)

5. Übersicht über Englisch (US):
   - Englisch (Jamaika)
   - Englisch (Karibik)
6. Übersicht über Lateinamerika (Lat):
   - Spanisch (Mexiko)
   - Spanisch (Guatemala)
   - Spanisch (Costa Rica)
   - Spanisch (Panama)
   - Spanisch (Dominikanische Republik)
   - Spanisch (Venezuela)
   - Spanisch (Kolumbien)
   - Spanisch (Peru)
   - Spanisch (Argentinien)
   - Spanisch (Ecuador)
   - Spanisch (Chile)
   - Spanisch (Uruguay)
   - Spanisch (Paraguay)
   - Spanisch (Bolivien)
7. Die folgenden indischen Schriften werden durch Unicode unterstützt: Hindi, Gujarati, Kannada, Konkani, Marathi, Punjabi, Sanskrit, Tamil und Telugu.
8. Codepage 950 wird auch als Big5 bezeichnet. Die Microsoft-Codepage 950 unterscheidet sich von der IBM Codepage 950 in folgenden Punkten:

Bereich	**Beschreibung**	**IBM**	**Microsoft**	**Unterschied**
X'8140' - X'8DFE'	Benutzerdefinierte Zeichen	Benutzer-definierter Bereich	Benutzer-definierter Bereich	Kein Unterschied
X'8E40' - X'A0FE'	Benutzerdefinierte Zeichen	Benutzer-definierter Bereich	Benutzer-definierter Bereich	Kein Unterschied
X'A140' - X'A3BF'	Sondersymbole	Systemzeichen	Systemzeichen	Kein Unterschied
X'A3C0' - X'A3E0'	Steuersymbole	Systemzeichen	Leer	Unterschied
X'A3E1' - X'A3FE'	Reserviert	Leer	Leer	Kein Unterschied
X'A440' - X'C67E'	Primäre Gebrauchszeichen	Systemzeichen	Systemzeichen	Kein Unterschied
X'C6A1' - X'C878'	Hinzugefügte Eten-Symbole	Systemzeichen	Benutzer-definierter Bereich	Unterschied
X'C879' - X'C8CC'	Hinzugefügte Eten-Symbole	Leer	Benutzer-definierter Bereich	Unterschied
X'C8CD' - X'C8D3'	Hinzugefügte Eten-Symbole	Systemzeichen	Benutzer-definierter Bereich	Unterschied
X'C8D4' - X'C8FD'	Reserviert	Systemzeichen	Benutzer-definierter Bereich	Unterschied

Bereich	Beschreibung	IBM	Microsoft	Unterschied
X'C8FE'	Ungültiges/ nicht definiertes Zeichen	Systemzeichen	Benutzer-definierter Bereich	Unterschied
X'C940' - X'F9D5'	Sekundäre Gebrauchszeichen	Systemzeichen	Systemzeichen	Kein Unterschied
X'F9D6' - X'F9FE'	Eten-Erweiterung für Big-5	Benutzer-definierter Bereich	Systemzeichen	Unterschied
X'FA40' - X'FEFE'	Benutzerdefinierte Zeichen	Benutzer-definierter Bereich	Benutzer-definierter Bereich	Kein Unterschied
X'8181' - X'8C82'	Benutzerdefinierte Zeichen	Benutzer-definierter Bereich	Leer	Unterschied
X'F286' - X'F9A0'	IBM Auswahl-zeichen	Systemzeichen	Leer	Unterschied
Gesamtanzahl Zeichen		14 060	13 502	
Gesamtanzahl benutzerdefinierter Zeichen		6 204	6 217	
Gesamtanzahl definierter Code-punkte		20 264	19 719	

Zugehörige Tasks:

- „Installieren der früheren Tabellen zur Konvertierung zwischen Codepage 1394 und Unicode" auf Seite 316

Aktivieren und Inaktivieren der Unterstützung für das Euro-Symbol

DB2 Universal Database™ (DB2 UDB) stellt eine Unterstützung für das Euro-Währungssymbol bereit. Das Euro-Symbol wurde zahlreichen Codepages hinzugefügt. Viele interne Codepagekonvertierungstabellen von DB2 UDB sowie viele externe Dateien mit Codepagekonvertierungstabellen, die sich im Verzeichnis `sqllib/conv/` befinden, wurden zur Unterstützung des Euro-Symbols erweitert.

ANSI-Codepages von Microsoft wurden modifiziert, um das Euro-Währungssymbol an Position X'80' einzufügen. Die Codepage 850 wurde dergestalt modifiziert, dass das Zeichen I OHNE PUNKT (an Position X'D5') durch das Euro-Währungssymbol ersetzt wurde. Interne Codepagekonvertierungsroutinen von DB2 UDB verwenden diese überarbeiteten Codepagedefinitionen standardmäßig, um die Unterstützung für das Euro-Symbol bereitzustellen.

Wenn Sie jedoch die Definitionen der Codepagekonvertierungstabellen ohne Euro-Symbol verwenden wollen, führen Sie nach Abschluss der Installation die nachfolgende Prozedur aus.

Vorbereitung:

Zum Ersetzen vorhandener externer Dateien mit Codepagekonvertierungstabellen empfiehlt es sich, die aktuellen Dateien zu sichern, bevor Sie die Versionen ohne Euro-Symbol über diese Dateien kopieren.

Die Dateien befinden sich im Verzeichnis `sqllib/conv/`. Unter UNIX ist das Verzeichnis `sqllib/conv` mit dem Installationspfad von DB2 UDB durch eine Programmverbindung (Link) verbunden.

Vorgehensweise:

Gehen Sie wie folgt vor, um die Unterstützung für das Euro-Symbol zu inaktivieren:

1. Stoppen Sie das DB2 UDB-Exemplar.
2. Laden Sie die entsprechenden Konvertierungstabellendateien binär von folgenden Adressen herunter:
   - Für Big-Endian-Plattformen von ftp://ftp.software.ibm.com/ps/products/db2/info/vr8/conv/BigEndian/. Dieser FTP-Server stellt einen anonymen Zugriff bereit. Wenn Sie die Verbindung über die Befehlszeile herstellen, melden Sie sich dementsprechend als "anonymous" an und benutzen Sie Ihre E-Mail-Adresse als Kennwort. Wechseln Sie nach erfolgter Anmeldung in das Verzeichnis für Konvertierungstabellen: cd ps/products/db2/info/vr8/conv/BigEndian/
   - Für Little-Endian-Plattformen von ftp://ftp.software.ibm.com/ps/products/db2/info/vr8/conv/LittleEndian/. Dieser FTP-Server stellt einen anonymen Zugriff bereit. Wenn Sie die Verbindung über die Befehlszeile herstellen, melden Sie sich dementsprechend als "anonymous" an und benutzen Sie Ihre E-Mail-Adresse als Kennwort. Wechseln Sie nach erfolgter Anmeldung in das Verzeichnis für Konvertierungstabellen: cd ps/products/db2/info/vr8/conv/LittleEndian
3. Kopieren Sie die Dateien in Ihr Verzeichnis `sqllib/conv/`.
4. Starten Sie das DB2 UDB-Exemplar erneut.

Die Codepages 819 und 1047:

Die um das Euro-Symbol erweiterten Ersatzcodepages 923 (ISO 8859-15 Latin 9 ASCII) und 924 (Latin 9 Open System EBCDIC) für die Codepages 819 (ISO 8859-1 Latin 1 ASCII) bzw. 1047 (Latin 1 Open System EBCDIC) enthalten neben dem Euro-Symbol noch einige weitere neue Zeichen. DB2 UDB verwendet standardmäßig auch weiterhin die alten Definitionen (ohne Euro-Symbol) dieser beiden Codepages und der Konvertierungstabellen, also 819 und 1047. Sie haben zwei Möglichkeiten, die neue Codepage 923/924 und die zugeordneten Konvertierungstabellen zu aktivieren:

- Erstellen Sie eine neue Datenbank, die mit der neuen Codepage arbeitet. Zum Beispiel:

  ```
  DB2 CREATE DATABASE dbname USING CODESET ISO8859-15 TERRITORY US
  ```

- Kopieren Sie die Konvertierungstabellendatei 923 oder 924 im Verzeichnis `sqllib/conv/` in 819 bzw. 1047 oder benennen Sie sie entsprechend um.

Zugehörige Konzepte:

- „Character conversion" in *SQL Reference, Volume 1*

Zugehörige Referenzen:

- „Konvertierungstabellen für die Codepages 923 und 924" auf Seite 295
- „Konvertierungstabellendateien für Euro-fähige Codepages" auf Seite 288

Konvertierungstabellendateien für Euro-fähige Codepages

In den folgenden Tabellen sind die Konvertierungstabellen aufgelistet, die zur Unterstützung des Euro-Symbols erweitert wurden. Wenn Sie die Unterstützung für das Euro-Symbol inaktivieren wollen, laden Sie die Konvertierungstabellendatei herunter, die in der Spalte 'Konvertierungstabellendateien' angegeben ist.

Arabisch:

Datenbankserver-CCSIDs/CPGIDs	Datenbankclient-CCSIDs/CPGIDs	Konvertierungstabellendateien
420, 16804	1046, 9238	04201046.cnv, IBM00420.ucs
420, 16804	1256, 5352	04201256.cnv, IBM00420.ucs
420, 16804	1200, 1208, 13488, 17584	IBM00420.ucs
864, 17248	1046, 9238	08641046.cnv, 10460864.cnv, IBM00864.ucs
864, 17248	1256, 5352	08641256.cnv, 12560864.cnv, IBM00864.ucs
864, 17248	1200, 1208, 13488, 17584	IBM00864.ucs
1046, 9238	864, 17248	10460864.cnv, 08641046.cnv, IBM01046.ucs
1046, 9238	1089	10461089.cnv, 10891046.cnv, IBM01046.ucs
1046, 9238	1256, 5352	10461256.cnv, 12561046.cnv, IBM01046.ucs
1046, 9238	1200, 1208, 13488, 17584	IBM01046.ucs
1089	1046, 9238	10891046.cnv, 10461089.cnv
1256, 5352	864, 17248	12560864.cnv, 08641256.cnv, IBM01256.ucs
1256, 5352	1046, 9238	12561046.cnv, 10461256.cnv, IBM01256.ucs
1256, 5352	1200, 1208, 13488, 17584	IBM01256.ucs

Baltikum:

Datenbankserver-CCSIDs/CPGIDs	Datenbankclient-CCSIDs/CPGIDs	Konvertierungstabellendateien
921, 901	1257	09211257.cnv, 12570921.cnv, IBM00921.ucs
921, 901	1200, 1208, 13488, 17584	IBM00921.ucs
1112, 1156	1257, 5353	11121257.cnv
1257, 5353	921, 901	12570921.cnv, 09211257.cnv, IBM01257.ucs
1257, 5353	922, 902	12570922.cnv, 09221257.cnv, IBM01257.ucs
1257, 5353	1200, 1208, 13488, 17584	IBM01257.ucs

Weißrussland:

Datenbankserver-CCSIDs/CPGIDs	Datenbankclient-CCSIDs/CPGIDs	Konvertierungstabellendateien
1131, 849	1251, 5347	11311251.cnv, 12511131.cnv
1131, 849	1283	11311283.cnv

Kyrillisch:

Datenbankserver-CCSIDs/CPGIDs	Datenbankclient-CCSIDs/CPGIDs	Konvertierungstabellendateien
855, 872	866, 808	08550866.cnv, 08660855.cnv
855, 872	1251, 5347	08551251.cnv, 12510855.cnv
866, 808	855, 872	08660855.cnv, 08550866.cnv
866, 808	1251, 5347	08661251.cnv, 12510866.cnv
1025, 1154	855, 872	10250855.cnv, IBM01025.ucs
1025, 1154	866, 808	10250866.cnv, IBM01025.ucs
1025, 1154	1131, 849	10251131.cnv, IBM01025.ucs
1025, 1154	1251, 5347	10251251.cnv, IBM01025.ucs
1025, 1154	1200, 1208, 13488, 17584	IBM01025.ucs
1251, 5347	855, 872	12510855.cnv, 08551251.cnv, IBM01251.ucs
1251, 5347	866, 808	12510866.cnv, 08661251.cnv, IBM01251.ucs
1251, 5347	1124	12511124.cnv, 11241251.cnv, IBM01251.ucs
1251, 5347	1125, 848	12511125.cnv, 11251251.cnv, IBM01251.ucs
1251, 5347	1131, 849	12511131.cnv, 11311251.cnv, IBM01251.ucs
1251, 5347	1200, 1208, 13488, 17584	IBM01251.ucs

Estland:

Datenbankserver-CCSIDs/CPGIDs	Datenbankclient-CCSIDs/CPGIDs	Konvertierungstabellendateien
922, 902	1257	09221257.cnv, 12570922.cnv, IBM00922.ucs
922, 902	1200, 1208, 13488, 17584	IBM00922.ucs
1122, 1157	1257, 5353	11221257.cnv

Griechisch:

Datenbankserver-CCSIDs/CPGIDs	Datenbankclient-CCSIDs/CPGIDs	Konvertierungstabellendateien
423	869, 9061	04230869.cnv
813, 4909	869, 9061	08130869.cnv, 08690813.cnv, IBM00813.ucs

Datenbankserver-CCSIDs/CPGIDs	Datenbankclient-CCSIDs/CPGIDs	Konvertierungstabellendateien
813, 4909	1253, 5349	08131253.cnv, 12530813.cnv, IBM00813.ucs
813, 4909	1200, 1208, 13488, 17584	IBM00813.ucs
869, 9061	813, 4909	08690813.cnv, 08130869.cnv
869, 9061	1253, 5349	08691253.cnv, 12530869.cnv
875, 4971	813, 4909	08750813.cnv, IBM00875.ucs
875, 4971	1253, 5349	08751253.cnv, IBM00875.ucs
875, 4971	1200, 1208, 13488, 17584	IBM00875.ucs
1253, 5349	813, 4909	12530813.cnv, 08131253.cnv, IBM01253.ucs
1253, 5349	869, 9061	12530869.cnv, 08691253.cnv, IBM01253.ucs
1253, 5349	1200, 1208, 13488, 17584	IBM01253.ucs

Hebräisch:

Datenbankserver-CCSIDs/CPGIDs	Datenbankclient-CCSIDs/CPGIDs	Konvertierungstabellendateien
424, 12712	856, 9048	04240856.cnv, IBM00424.ucs
424, 12712	862, 867	04240862.cnv, IBM00424.ucs
424, 12712	916	04240916.cnv, IBM00424.ucs
424, 12712	1255, 5351	04241255.cnv, IBM00424.ucs
424, 12712	1200, 1208, 13488, 17584	IBM00424.ucs
856, 9048	862, 867	08560862.cnv, 08620856.cnv, IBM0856.ucs
856, 9048	916	08560916.cnv, 09160856.cnv, IBM0856.ucs
856, 9048	1255, 5351	08561255.cnv, 12550856.cnv, IBM0856.ucs
856, 9048	1200, 1208, 13488, 17584	IBM0856.ucs
862, 867	856, 9048	08620856.cnv, 08560862.cnv, IBM00862.ucs
862, 867	916	08620916.cnv, 09160862.cnv, IBM00862.ucs
862, 867	1255, 5351	08621255.cnv, 12550862.cnv, IBM00862.ucs
862, 867	1200, 1208, 13488, 17584	IBM00862.ucs
916	856, 9048	09160856.cnv, 08560916.cnv
916	862, 867	09160862.cnv, 08620916.cnv
1255, 5351	856, 9048	12550856.cnv, 08561255.cnv, IBM01255.ucs
1255, 5351	862, 867	12550862.cnv, 08621255.cnv, IBM01255.ucs
1255, 5351	1200, 1208, 13488, 17584	IBM01255.ucs

Japanisch:

Datenbankserver-CCSIDs/CPGIDs	Datenbankclient-CCSIDs/CPGIDs	Konvertierungstabellendateien
290, 8482	850, 858	02900850.cnv
1027, 5123	850, 858	10270850.cnv

Latin-1:

Datenbankserver-CCSIDs/CPGIDs	Datenbankclient-CCSIDs/CPGIDs	Konvertierungstabellendateien
37, 1140	437	00370437.cnv, IBM00037.ucs
37, 1140	850, 858	00370850.cnv, IBM00037.ucs
37, 1140	860	00370860.cnv, IBM00037.ucs
37, 1140	1051	00371051.cnv, IBM00037.ucs
37, 1140	1252, 5348	00371252.cnv, IBM00037.ucs
37, 1140	1275	00371275.cnv, IBM00037.ucs
37, 1140	1200, 1208, 13488, 17584	IBM00037.ucs
273, 1141	437	02730437.cnv, IBM00273.ucs
273, 1141	850, 858	02730850.cnv, IBM00273.ucs
273, 1141	1051	02731051.cnv, IBM00273.ucs
273, 1141	1252, 5348	02731252.cnv, IBM00273.ucs
273, 1141	1275	02731275.cnv, IBM00273.ucs
273, 1141	1200, 1208, 13488, 17584	IBM00273.ucs
277, 1142	437	02770437.cnv, IBM00277.ucs
277, 1142	850, 858	02770850.cnv, IBM00277.ucs
277, 1142	1051	02771051.cnv, IBM00277.ucs
277, 1142	1252, 5348	02771252.cnv, IBM00277.ucs
277, 1142	1275	02771275.cnv, IBM00277.ucs
277, 1142	1200, 1208, 13488, 17584	IBM00277.ucs
278, 1143	437	02780437.cnv, IBM00278.ucs
278, 1143	850, 858	02780850.cnv, IBM00278.ucs
278, 1143	1051	02781051.cnv, IBM00278.ucs
278, 1143	1252, 5348	02781252.cnv, IBM00278.ucs
278, 1143	1275	02781275.cnv, IBM00278.ucs
278, 1143	1200, 1208, 13488, 17584	IBM00278.ucs
280, 1144	437	02800437.cnv, IBM00280.ucs
280, 1144	850, 858	02800850.cnv, IBM00280.ucs
280, 1144	1051	02801051.cnv, IBM00280.ucs
280, 1144	1252, 5348	02801252.cnv, IBM00280.ucs
280, 1144	1275	02801275.cnv, IBM00280.ucs
280, 1144	1200, 1208, 13488, 17584	IBM00280.ucs
284, 1145	437	02840437.cnv, IBM00284.ucs
284, 1145	850, 858	02840850.cnv, IBM00284.ucs

Datenbankserver-CCSIDs/CPGIDs	Datenbankclient-CCSIDs/CPGIDs	Konvertierungstabellendateien
284, 1145	1051	02841051.cnv, IBM00284.ucs
284, 1145	1252, 5348	02841252.cnv, IBM00284.ucs
284, 1145	1275	02841275.cnv, IBM00284.ucs
284, 1145	1200, 1208, 13488, 17584	IBM00284.ucs
285, 1146	437	02850437.cnv, IBM00285.ucs
285, 1146	850, 858	02850850.cnv, IBM00285.ucs
285, 1146	1051	02851051.cnv, IBM00285.ucs
285, 1146	1252, 5348	02851252.cnv, IBM00285.ucs
285, 1146	1275	02851275.cnv, IBM00285.ucs
285, 1146	1200, 1208, 13488, 17584	IBM00285.ucs
297, 1147	437	02970437.cnv, IBM00297.ucs
297, 1147	850, 858	02970850.cnv, IBM00297.ucs
297, 1147	1051	02971051.cnv, IBM00297.ucs
297, 1147	1252, 5348	02971252.cnv, IBM00297.ucs
297, 1147	1275	02971275.cnv, IBM00297.ucs
297, 1147	1200, 1208, 13488, 17584	IBM00297.ucs
437	850, 858	04370850.cnv, 08500437.cnv
500, 1148	437	05000437.cnv, IBM00500.ucs
500, 1148	850, 858	05000850.cnv, IBM00500.ucs
500, 1148	857, 9049	05000857.cnv, IBM00500.ucs
500, 1148	920	05000920.cnv, IBM00500.ucs
500, 1148	1051	05001051.cnv, IBM00500.ucs
500, 1148	1114, 5210	05001114.cnv, IBM00500.ucs
500, 1148	1252, 5348	05001252.cnv, IBM00500.ucs
500, 1148	1254, 5350	05001254.cnv, IBM00500.ucs
500, 1148	1275	05001275.cnv, IBM00500.ucs
500, 1148	1200, 1208, 13488, 17584	IBM00500.ucs
850, 858	437	08500437.cnv, 04370850.cnv
850, 858	860	08500860.cnv, 08600850.cnv
850, 858	1114, 5210	08501114.cnv, 11140850.cnv
850, 858	1275	08501275.cnv, 12750850.cnv
860	850, 858	08600850.cnv, 08500860.cnv
871, 1149	437	08710437.cnv, IBM00871.ucs
871, 1149	850, 858	08710850.cnv, IBM00871.ucs
871, 1149	1051	08711051.cnv, IBM00871.ucs
871, 1149	1252, 5348	08711252.cnv, IBM00871.ucs
871, 1149	1275	08711275.cnv, IBM00871.ucs
871, 1149	1200, 1208, 13488, 17584	IBM00871.ucs
1275	850, 858	12750850.cnv, 08501275.cnv

Latin-2:

Datenbankserver-CCSIDs/CPGIDs	Datenbankclient-CCSIDs/CPGIDs	Konvertierungstabellendateien
852, 9044	1250, 5346	08521250.cnv, 12500852.cnv
870, 1153	852, 9044	08700852.cnv, IBM00870.ucs
870, 1153	1250, 5346	08701250.cnv, IBM00870.ucs
870, 1153	1200, 1208, 13488, 17584	IBM00870.ucs
1250, 5346	852, 9044	12500852.cnv, 08521250.cnv, IBM01250.ucs
1250, 5346	1200, 1208, 13488, 17584	IBM01250.ucs

Vereinfachtes Chinesisch:

Datenbankserver-CCSIDs/CPGIDs	Datenbankclient-CCSIDs/CPGIDs	Konvertierungstabellendateien
837, 935, 1388	1200, 1208, 13488, 17584	1388ucs2.cnv
1386	1200, 1208, 13488, 17584	1386ucs2.cnv, ucs21386.cnv

Traditionelles Chinesisch:

Datenbankserver-CCSIDs/CPGIDs	Datenbankclient-CCSIDs/CPGIDs	Konvertierungstabellendateien
937, 835, 1371	950, 1370	09370950.cnv, 0937ucs2.cnv
937, 835, 1371	1200, 1208, 13488, 17584	0937ucs2.cnv
1114, 5210	850, 858	11140850.cnv, 08501114.cnv

Thailand:

Datenbankserver-CCSIDs/CPGIDs	Datenbankclient-CCSIDs/CPGIDs	Konvertierungstabellendateien
838, 1160	874, 1161	08380874.cnv, IBM00838.ucs
838, 1160	1200, 1208, 13488, 17584	IBM00838.ucs
874, 1161	1200, 1208, 13488, 17584	IBM00874.ucs

Türkisch:

Datenbankserver-CCSIDs/CPGIDs	Datenbankclient-CCSIDs/CPGIDs	Konvertierungstabellendateien
857, 9049	1254, 5350	08571254.cnv, 12540857.cnv
1026, 1155	857, 9049	10260857.cnv, IBM01026.ucs
1026, 1155	1254, 5350	10261254.cnv, IBM01026.ucs
1026, 1155	1200, 1208, 13488, 17584	IBM01026.ucs
1254, 5350	857, 9049	12540857.cnv, 08571254.cnv, IBM01254.ucs
1254, 5350	1200, 1208, 13488, 17584	IBM01254.ucs

Ukraine:

Datenbankserver-CCSIDs/CPGIDs	Datenbankclient-CCSIDs/CPGIDs	Konvertierungstabellendateien
1123, 1158	1124	11231124.cnv
1123, 1158	1125, 848	11231125.cnv
1123, 1158	1251, 5347	11231251.cnv
1124	1251, 5347	11241251.cnv, 12511124.cnv
1125, 848	1251, 5347	11251251.cnv, 12511125.cnv

Unicode:

Datenbankserver-CCSIDs/CPGIDs	Datenbankclient-CCSIDs/CPGIDs	Konvertierungstabellendateien
1200, 1208, 13488, 17584	813, 4909	IBM00813.ucs
1200, 1208, 13488, 17584	862, 867	IBM00862.ucs
1200, 1208, 13488, 17584	864, 17248	IBM00864.ucs
1200, 1208, 13488, 17584	874, 1161	IBM00874.ucs
1200, 1208, 13488, 17584	921, 901	IBM00921.ucs
1200, 1208, 13488, 17584	922, 902	IBM00922.ucs
1200, 1208, 13488, 17584	1046, 9238	IBM01046.ucs
1200, 1208, 13488, 17584	1250, 5346	IBM01250.ucs
1200, 1208, 13488, 17584	1251, 5347	IBM01251.ucs
1200, 1208, 13488, 17584	1253, 5349	IBM01253.ucs
1200, 1208, 13488, 17584	1254, 5350	IBM01254.ucs
1200, 1208, 13488, 17584	1255, 5351	IBM01255.ucs
1200, 1208, 13488, 17584	1256, 5352	IBM01256.ucs
1200, 1208, 13488, 17584	1386	ucs21386.cnv, 1386ucs2.cnv

Vietnamesisch:

Datenbankserver-CCSIDs/CPGIDs	Datenbankclient-CCSIDs/CPGIDs	Konvertierungstabellendateien
1130, 1164	1258, 5354	11301258.cnv
1258, 5354	1129, 1163	12581129.cnv

Zugehörige Konzepte:

- „Character conversion" in *SQL Reference, Volume 1*

Zugehörige Tasks:

- „Aktivieren und Inaktivieren der Unterstützung für das Euro-Symbol" auf Seite 286

Konvertierungstabellen für die Codepages 923 und 924

In der folgenden Liste sind alle Konvertierungstabellendateien für Codepages aufgeführt, die den Codepages 923 und 924 zugeordnet sind. Jeder Dateiname besitzt die Form XXXXYYYY.cnv oder ibmZZZZZ.ucs, wobei XXXXX die Nummer der Quellencodepage und YYYY die Nummer der Zielcodepage ist. Die Datei ibmZZZZZ.ucs unterstützt die Konvertierung zwischen Codepage ZZZZZ und Unicode.

Zur Aktivierung einer bestimmten Codepagekonvertierungstabelle benennen Sie die Konvertierungstabellendatei in den neuen Namen um (oder kopieren die Datei in den neuen Namen), der in der zweiten Spalte gezeigt ist.

Wenn Sie zum Beispiel bei der Verbindung eines Clients mit 8859-1/15 (Latin 1/9) mit einer Windows-1252-Datenbank das Euro-Symbol verwenden wollen, müssen Sie die folgenden Dateien mit Codepagekonvertierungstabellen im Verzeichnis sqllib/conv/ in den neuen Namen umbenennen bzw. kopieren:

- 09231252.cnv in 08191252.cnv
- 12520923.cnv in 12520819.cnv
- ibm00923.ucs in ibm00819.ucs

Konvertierungstabellendateien für 923 und 924 im Verzeichnis sqlllib/conv/ directory	**Neuer Name**
00370923.cnv	00370819.cnv
02730923.cnv	02730819.cnv
02770923.cnv	02770819.cnv
02780923.cnv	02780819.cnv
02800923.cnv	02800819.cnv
02840923.cnv	02840819.cnv
02850923.cnv	02850819.cnv
02970923.cnv	02970819.cnv
04370923.cnv	04370819.cnv
05000923.cnv	05000819.cnv
08500923.cnv	08500819.cnv
08600923.cnv	08600819.cnv
08630923.cnv	08630819.cnv
08710923.cnv	08710819.cnv
09230437.cnv	08190437.cnv
09230500.cnv	08190500.cnv
09230850.cnv	08190850.cnv
09230860.cnv	08190860.cnv
09230863.cnv	08190863.cnv
09231043.cnv	08191043.cnv
09231051.cnv	08191051.cnv
09231114.cnv	08191114.cnv
09231252.cnv	08191252.cnv

Konvertierungstabellendateien für 923 und 924 im Verzeichnis sqlllib/conv/ directory	Neuer Name
09231275.cnv	08191275.cnv
09241252.cnv	10471252.cnv
10430923.cnv	10430819.cnv
10510923.cnv	10510819.cnv
11140923.cnv	11140819.cnv
12520923.cnv	12520819.cnv
12750923.cnv	12750819.cnv
ibm00923.ucs	ibm00819.ucs

Zugehörige Konzepte:

- „Character conversion" in *SQL Reference, Volume 1*

Zugehörige Tasks:

- „Aktivieren und Inaktivieren der Unterstützung für das Euro-Symbol" auf Seite 286

Auswählen einer Sprache für Ihre Datenbank

Bei der Erstellung einer Datenbank müssen Sie entscheiden, in welcher Sprache die Daten gespeichert werden sollen. Wenn Sie eine Datenbank erstellen, können Sie das Gebiet (TERRITORY) und den codierten Zeichensatz (CODESET) angeben. Das Gebiet und und der codierte Zeichensatz können sich von den aktuellen Einstellungen des Betriebssystems unterscheiden. Werden bei der Erstellung einer Datenbank kein Gebiet und kein codierter Zeichensatz explizit angegeben, wird die Datenbank unter Verwendung der aktuellen Ländereinstellungen (Locale) erstellt. Stellen Sie bei der Auswahl eines codierten Zeichensatzes sicher, dass mit diesem Zeichensatz alle Zeichen der Sprache codiert werden können, die Sie verwenden wollen.

Eine alternative Option besteht darin, die Daten in einer Unicode-Datenbank zu speichern. In diesem Fall brauchen Sie keine bestimmte Sprache auszuwählen, da die Unicode-Codierung die Zeichen annähernd aller lebendigen Sprachen der Welt umfasst.

Ländereinstellung für den DB2-Verwaltungsserver

Stellen Sie sicher, dass die Ländereinstellung (Locale) des DB2-Verwaltungsserverexemplars mit der Ländereinstellung des DB2 Universal Database™-Exemplars (DB2 UDB) kompatibel ist. Ansonsten kann das DB2 UDB-Exemplar nicht mit dem DB2-Verwaltungsserver kommunizieren.

Wenn die Umgebungsvariable LANG im Benutzerprofil des DB2-Verwaltungsservers nicht definiert ist, wird der DB2-Verwaltungsserver mit der Standardländereinstellung des Systems gestartet. Falls die Standardländereinstellung des Systems nicht definiert ist, wird der DB2-Verwaltungsserver mit Codepage 819 gestartet. Wenn das DB2 UDB-Exemplar mit einer der DBCS-Ländereinstellungen arbeitet und der DB2-Verwaltungsserver mit Codepage 819 gestartet wird, ist das Exemplar nicht in der Lage, mit dem DB2-Verwaltungsserver zu kommunizieren. Die Ländereinstellung des DB2-Verwaltungsservers und die Ländereinstellung des DB2 UDB-Exemplars müssen kompatibel sein.

Zum Beispiel sollte für ein Linux-System mit vereinfachtem Chinesisch im Benutzerprofil des DB2-Verwaltungsservers die Variable `LANG=zh_CN` definiert sein.

Aktivieren der Unterstützung für bidirektionale Zeichensätze

Bidirektionale Layoutumsetzungen werden in DB2 Universal Database mit den neuen Definitionen der IDs für codierte Zeichensätze (CCSIDs - Coded Character Set Identifiers) implementiert. Für die neuen spezifischen CCSIDs für bidirektionale Zeichensätze werden Layoutumsetzungen entweder anstatt oder zusätzlich zur Umsetzung von Codepages durchgeführt. Damit diese Unterstützung verwendet werden kann, muss die Registriervariable DB2BIDI auf YES gesetzt werden. Standardmäßig ist diese Variable nicht gesetzt. Sie wird vom Server für alle Umsetzungen verwendet und kann nur gesetzt werden, wenn der Server gestartet ist. Das Einstellen von DB2BIDI auf YES kann sich wegen der zusätzlichen Prüfung und der Layoutumsetzungen auf die Leistung auswirken.

Einschränkungen:

Es gelten die folgenden Einschränkungen:

- Wenn Sie eine CCSID auswählen, die für die Codepage oder den Zeichenfolgetyp Ihrer Clientplattform nicht geeignet ist, erhalten Sie möglicherweise unerwartete Ergebnisse. Wenn Sie eine inkompatible CCSID (z. B. CCSID für Hebräisch zur Verbindung zu einer arabischen Datenbank) verwenden oder wenn DB2BIDI nicht für den Server definiert wurde, erhalten Sie eine Fehlernachricht, wenn Sie versuchen, die Verbindung herzustellen.
- Der Befehlszeilenprozessor von DB2 Universal Database™ (DB2 UDB) im Window-Betriebssystem besitzt keine bidirektionale Unterstützung.
- Die CCSID-Überschreibung wird nicht unterstützt, wenn die HOST-EBCDIC-Plattform der Client und DB2 UDB der Server ist.

Bei der Umsetzung von einer arabischen CCSID in eine andere arabische CCSID verfährt DB2 UDB nach der folgenden Logik, um die LamAlef-Ligatur zu trennen. Die Ligaturentrennung erfolgt, wenn das Attribut für Textgestaltung (Text Shaping) der arabischen Quellen-CCSID den Wert SHAPED, das der arabischen Ziel-CCSID jedoch den Wert UNSHAPED hat.

Die folgende Logik zur Trennung der LamAlef-Ligatur wird angewendet:

1. Wenn das *letzte* Zeichen des Datenstroms ein Leerzeichen ist, wird jedes Zeichen nach der LamAlef-Ligatur an das Ende des Datenstroms verschoben, so dass ein freier Bereich für die aktuelle LamAlef-Ligatur verfügbar wird, die in die beiden konstitutiven Zeichen aufzutrennen ist: Lam und Alef.
2. Wenn andererseits das *erste* Zeichen des Datenstroms ein Leerzeichen ist, wird jedes Zeichen vor der LamAlef-Ligatur an den Anfang des Datenstroms verschoben, so dass ein freier Bereich für die aktuelle LamAlef-Ligatur verfügbar wird, die in die beiden konstitutiven Zeichen aufzutrennen ist: Lam und Alef.
3. Anderenfalls gibt es kein Leerzeichen am Anfang und am Ende des Datenstroms und die LamAlef-Ligatur kann nicht aufgetrennt werden. Wenn die Ziel-CCSID die LamAlef-Ligatur enthält, bleibt die LamAlef-Ligatur, wie sie ist. Ansonsten wird die LamAlef-Ligatur durch das Substitutionszeichen der Ziel-CCSID ersetzt.

Bei der umgekehrten Konvertierung von einer arabischen CCSID mit dem Textgestaltungsattribut UNSHAPED in eine arabische CCSID mit dem Textgestaltungattribut SHAPED werden die Quellenzeichen Lam und Alef zu einem Ligaturzeichen kontrahiert und ein Leerzeichen am Ende des Datenstroms des Zielbereichs eingefügt.

Vorgehensweise:

Gehen Sie wie folgt vor, um eine bestimmte bidirektionale CCSID in einer Nicht-DRDA-Umgebung anzugeben:

- Stellen Sie sicher, dass die Registriervariable DB2BIDI auf den Wert YES gesetzt ist.
- Wählen Sie die CCSID aus, die den Merkmalen Ihres Clients entspricht, und setzen Sie die Variable DB2CODEPAGE auf diesen Wert.
- Wenn Sie bereits eine Verbindung zur Datenbank haben, müssen Sie den Befehl TERMINATE absetzen und die Verbindung anschließend erneut herstellen, damit die neue Einstellung der Variablen DB2CODEPAGE wirksam wird.

Wenn bei DRDA-Umgebungen die HOST-EBCDIC-Plattform diese CCSIDs für bidirektionale Zeichensätze ebenfalls unterstützt, müssen Sie nur den Wert der Variablen DB2CODEPAGE definieren. Beachten Sie, dass Sie die gleiche CCSID nicht weiter im Parameter BIBI im Feld PARMS des DCS-Datenbankverzeichniseintrags für die Serverdatenbank angegeben dürfen. Ansonsten würde eine zusätzliche Bidi-Layoutumsetzung stattfinden, und alle arabischen Daten würden falsch zurückkonvertiert. Wenn die HOST-Plattform diese CCSIDs jedoch nicht unterstützt, müssen Sie eine CCSID-Überschreibung für den HOST-Datenbankserver angeben, zu dem Sie eine Verbindung herstellen. Dies wird durch die Verwendung des Parameters BIDI im Feld PARMS des DCS-Datenbankverzeichniseintrags für die Serverdatenbank erreicht. Die Überschreibung ist notwendig, da in einer DRDA-Umgebung Umsetzungen von Codepages und Layoutumsetzungen vom Empfänger der Daten durchgeführt werden. Wenn der HOST-Server diese CCSIDs jedoch nicht unterstützt, führt er keine Layoutumsetzung für die Daten durch, die er von DB2 UDB empfängt. Wenn Sie eine CCSID-Überschreibung verwenden, führt der Client unter DB2 UDB die Layoutumsetzung auch für die abgehenden Daten durch.

Zugehörige Konzepte:

- „Unterstützung für bidirektionale Zeichensätze mit DB2 Connect" auf Seite 301
- „Handhabung von BIDI-Daten" in *DB2 Connect Benutzerhandbuch*

Zugehörige Referenzen:

- „Spezifische CCSIDs für bidirektionale Zeichensätze" auf Seite 298
- „Allgemeine Registrierdatenbankvariablen" in *Systemverwaltung: Optimierung*

Spezifische CCSIDs für bidirektionale Zeichensätze

Die folgenden bidirektionalen Attribute sind erforderlich für die korrekte Behandlung bidirektionaler Daten auf unterschiedlichen Plattformen:

- Texttyp
- Numerische Gestaltung
- Ausrichtung
- Textgestaltung
- Symmetrische Spiegelung

Da die Standardeinstellungen auf den unterschiedlichen Plattformen nicht gleich sind, können Probleme auftreten, wenn DB2 UDB-Daten (DB2 Universal Database™) von einer Plattform auf eine andere übertragen werden. Das Windows-Betriebssystem verwendet beispielsweise Daten im Format LOGICAL UNSHAPED, während z/OS und OS/390 im Allgemeinen mit Daten im Format SHAPED VISUAL arbeiten. Daten, die ohne Unterstützung für bidirektionale Attribute von DB2 Universal Database für z/OS und OS/390 auf DB2 UDB unter 32-Bit-Windows-Betriebssysteme übertragen werden, können daher falsch angezeigt werden.

DB2 UDB unterstützt bidirektionale Datenattribute über spezielle bidirektionale IDs für codierte Zeichensätze (CCSIDs). Die folgenden CCSIDs für bidirektionale Zeichen wurden definiert und mit DB2 UDB implementiert, wie in Tabelle 95 gezeigt. CDRA-Zeichenfolgetypen sind wie in Tabelle 96 auf Seite 300 gezeigt definiert.

Tabelle 95. Bidirektionale CCSIDs

CCSID	Codepage	Zeichenfolgetyp
420	420	4
424	424	4
856	856	5
862	862	4
864	864	5
867	862	4
916	916	5
1046	1046	5
1089	1089	5
1200	1200	10
1208	1208	10
1255	1255	5
1256	1256	5
5351	1255	5
5352	1256	5
8612	420	5
8616	424	10
9048	856	5
9238	1046	5
12712	424	4
13488	13488	10
16804	420	4
17248	864	5
62208	856	4
62209	862	10
62210	916	4
62211	424	5
62213	862	5
62215	1255	4
62218	864	4

Tabelle 95. Bidirektionale CCSIDs (Forts.)

CCSID	Codepage	Zeichenfolgetyp
62220	856	6
62221	862	6
62222	916	6
62223	1255	6
62224	420	6
62225	864	6
62226	1046	6
62227	1089	6
62228	1256	6
62229	424	8
62230	856	8
62231	862	8
62232	916	8
62233	420	8
62234	420	9
62235	424	6
62236	856	10
62237	1255	8
62238	916	10
62239	1255	10
62240	424	11
62241	856	11
62242	862	11
62243	916	11
62244	1255	11
62245	424	10
62246	1046	8
62247	1046	9
62248	1046	4
62249	1046	12
62250	420	12

Tabelle 96. CDRA-Zeichenfolgetypen

Zeichenfolgetyp	Texttyp	Numerische Gestaltung	Ausrichtung	Textgestaltung	Symmetrische Spiegelung
4	Visual	Passthrough	LTR	Shaped	Off
5	Implicit	Arabic	LTR	Unshaped	On
6	Implicit	Arabic	RTL	Unshaped	On
7*	Visual	Passthrough	Contextual*	Unshaped ligature	Off

Tabelle 96. CDRA-Zeichenfolgetypen (Forts.)

Zeichenfolgetyp	Texttyp	Numerische Gestaltung	Ausrichtung	Textgestaltung	Symmetrische Spiegelung
8	Visual	Passthrough	RTL	Shaped	Off
9	Visual	Passthrough	RTL	Shaped	On
10	Implicit	Arabic	Contextual LTR	Unshaped	On
11	Implicit	Arabic	Contextual RTL	Unshaped	On
12	Implicit	Arabic	RTL	Shaped	Off

Anmerkung: * Die Zeichenfolgenausrichtung ist von links nach rechts (LTR), wenn das erste alphabetische Zeichen ein lateinischer Buchstabe ist, und von rechts nach links (RTL), wenn es ein arabisches oder hebräisches Zeichen ist. Zeichen sind UNSHAPED, LamAlef-Ligaturen werden jedoch beibehalten und nicht in Bestandteile getrennt.

Zugehörige Konzepte:

- „Unterstützung für bidirektionale Zeichensätze mit DB2 Connect" auf Seite 301

Zugehörige Tasks:

- „Aktivieren der Unterstützung für bidirektionale Zeichensätze" auf Seite 297

Unterstützung für bidirektionale Zeichensätze mit DB2 Connect

Wenn Daten zwischen DB2® Connect und einer Datenbank auf dem Server ausgetauscht werden, wird die Umsetzung der eingehenden Daten normalerweise von der Empfängermaschine durchgeführt. Dieselbe Konvention gilt normalerweise auch für bidirektionale Layoutumsetzungen und findet zusätzlich zur üblichen Codepageumsetzung statt. DB2 Connect™ kann optional bidirektionale Layoutumsetzungen an den Daten ausführen, die gerade an die Serverdatenbank gesendet werden, ebenso wie an den Daten, die von der Serverdatenbank empfangen werden.

Damit DB2 Connect bidirektionale Layoutumsetzungen an abgehenden Daten für eine Serverdatenbank ausführen kann, muss die bidirektionale CCSID der Serverdatenbank überschrieben werden. Dies wird durch die Verwendung des Parameters BIDI im Feld PARMS des DCS-Datenbankverzeichniseintrags für die Serverdatenbank erreicht.

Anmerkung: Wenn DB2 Connect eine Layoutumsetzung an den Daten ausführen soll, die gerade an die DB2 Universal Database™-Host- oder iSeries™-Datenbank gesendet werden, brauchen Sie zwar die CCSID nicht zu überschreiben, jedoch müssen Sie den Parameter BIDI dem Feld PARMS des DCS-Datenbankverzeichnisses hinzufügen. In diesem Fall sollten Sie als CCSID die Standard-CCSID der DB2 UDB-Host- oder iSeries-Datenbank angeben.

Der Parameter BIDI muss als neunter Parameter im Feld PARMS zusammen mit der bidirektionalen CCSID, mit der Sie die bidirektionale Standard-CCSID der

Serverdatenbank überschreiben wollen, angegeben werden:

```
",,,,,,,,BIDI=xyz"
```

Dabei ist *xyz* die CCSID-Überschreibung.

Anmerkung: Die Registriervariable DB2BIDI muss auf den Wert YES gesetzt werden, damit der Parameter BIDI wirksam werden kann.

Die Verwendung dieser Einrichtung lässt sich am besten an einem Beispiel erläutern.

Angenommen, Sie haben einen hebräischen DB2 UDB-Client, der CCSID 62213 (bidirektionaler Zeichenfolgetyp 5) ausführt, und Sie wollen auf eine DB2 UDB-Host- oder iSeries-Datenbank zugreifen, die CCSID 00424 (bidirektionaler Zeichenfolgetyp 4) ausführt. Sie wissen jedoch, dass die in der DB2 UDB-Host- oder iSeries-Datenbank enthaltenen Daten auf der CCSID 08616 (bidirektionaler Zeichenfolgetyp 6) basieren.

Hier stellen sich zwei Probleme: Das erste Problem besteht darin, dass die DB2 UDB-Host- oder iSeries-Datenbank den Unterschied zwischen den bidirektionalen Zeichenfolgetypen mit den CCSIDs 00424 und 08616 nicht kennt. Das zweite Problem ist, dass die DB2 UDB-Host- oder iSeries-Datenbank die DB2 UDB-Client-CCSID (62213) nicht erkennt. Sie unterstützt nur die CCSID 00862, die auf derselben Codepage wie CCSID 62213 basiert.

Sie müssen zunächst sicherstellen, dass die an die DB2 UDB-Host- oder iSeries-Datenbank gesendeten Daten das Format des bidirektionalen Zeichenfolgetyps 6 haben. Ferner müssen Sie DB2 Connect anweisen, eine bidirektionale Umsetzung der Daten auszuführen, die von der DB2 UDB-Host- oder iSeries-Datenbank empfangen werden. Sie müssen den folgenden Katalogisierungsbefehl für die DB2 UDB-Host- oder iSeries-Datenbank verwenden:

```
db2 catalog dcs database nydb1 as telaviv parms ",,,,,,,,BIDI=08616"
```

Dieser Befehl weist DB2 Connect an, die CCSID 00424 der DB2 UDB-Host- oder iSeries-Datenbank mit der CCSID 08616 zu überschreiben. Diese Überschreibung wird wie folgt verarbeitet:

1. DB2 Connect stellt eine Verbindung zur DB2 UDB-Host- oder iSeries-Datenbank mit der CCSID 00862 her.
2. DB2 Connect führt eine bidirektionale Layoutumsetzung der Daten aus, die an die DB2 UDB-Host- oder iSeries-Datenbank *gesendet* werden sollen. Die CCSID 62213 (bidirektionaler Zeichenfolgetyp 5) wird in die CCSID 62221 (bidirektionaler Zeichenfolgetyp 6) umgesetzt.
3. DB2 Connect führt eine bidirektionale Layoutumsetzung der Daten aus, die von der DB2 UDB-Host- oder iSeries-Datenbank *empfangen* werden. Diese Umsetzung erfolgt von der CCSID 08616 (bidirektionaler Zeichenfolgetyp 6) in die CCSID 62213 (bidirektionaler Zeichenfolgetyp 5).

Anmerkung: In einigen Fällen kann die Verwendung einer bidirektionalen CCSID die SQL-Abfrage selbst so ändern, dass sie nicht mehr vom DB2 UDB-Server erkannt wird. Sie sollten vor allem die Verwendung von CCSIDs mit IMPLICIT CONTEXTUAL und IMPLICIT RIGHT-TO-LEFT vermeiden, wenn ein anderer Zeichenfolgetyp verwendet werden kann. CONTEXTUAL-CCSIDs können zu unvorhersehbaren Ergebnissen führen, wenn die SQL-Abfrage Zeichenfolgen in Anfüh-

rungszeichen enthält. Vermeiden Sie daher in SQL-Anweisungen Zeichenfolgen in Anführungszeichen. Verwenden Sie nach Möglichkeit überall Hostvariablen.

Wenn eine bestimmte bidirektionale CCSID Probleme hervorruft, die nicht durch das Befolgen dieser Empfehlungen korrigiert werden können, setzen Sie die Variable DB2BIDI auf den Wert NO.

Zugehörige Konzepte:

- „Handhabung von BIDI-Daten" in *DB2 Connect Benutzerhandbuch*

Zugehörige Referenzen:

- „Spezifische CCSIDs für bidirektionale Zeichensätze" auf Seite 298

Sortierfolgen

Der Datenbankmanager vergleicht Zeichendaten mit Hilfe einer *Sortierfolge*. Dies ist eine Reihenfolge für eine Zeichengruppe, mit der festgelegt wird, ob ein bestimmtes Zeichen vor, nach oder gleichwertig mit einem anderen angeordnet wird.

Anmerkung: Daten aus Zeichenfolgen, die mit dem Attribut FOR BIT DATA definiert wurden, und BLOB-Daten werden mit der Binärsortierfolge sortiert.

Mit einer Sortierfolge kann beispielsweise angegeben werden, dass Großbuchstaben und Kleinbuchstaben gleich bewertet werden sollen.

Der Datenbankmanager ermöglicht das Erstellen von Datenbanken mit benutzerdefinierten Sortierfolgen. Für Unicode-Datenbanken werden die verschiedenen unterstützten Sortierfolgen im Abschnitt „Unicode-Implementierung in DB2 Universal Database" beschrieben. Die folgenden Abschnitte bieten Informationen, die Sie bei der Bestimmung und Implementierung einer bestimmten Sortierfolge für eine Datenbank unterstützen.

Jedes Einzelbytezeichen in einer Datenbank wird intern als eine eindeutige Nummer zwischen 0 und 255 (in Hexadezimalnotation X'00' und X'FF') dargestellt. Diese Nummer wird als *Codepunkt* des Zeichens bezeichnet. Die Zuordnung von Nummern zu Zeichen in einem Zeichensatz wird als Ganzes als *Codepage* bezeichnet. Eine Sortierfolge ist eine Zuordnung zwischen dem Codepunkt und der gewünschten Position jedes Zeichens in einer sortierten Folge. Der numerische Wert der Position wird als *Wertigkeit* des Zeichens in der Sortierfolge bezeichnet. Im einfachsten Fall einer Sortierfolge sind die Wertigkeiten mit den Codepunkten identisch. Eine solche Folge wird als *Identitätssortierfolge* bezeichnet.

Nehmen Sie zum Beispiel an, dass die Zeichen B und b die Codepunkte X'42' bzw. X'62' besitzen. Wenn sie (gemäß der Sortierfolgetabelle) beide die Sortierwertigkeit X'42' (B) haben, werden sie beim Sortieren als gleichwertig behandelt. Wenn die Sortierwertigkeit für B gleich X'9E' und die Sortierwertigkeit für b gleich X'9D' ist, wird b vor B sortiert. Die Sortierfolgetabelle gibt die Wertigkeit jedes Zeichens an. Diese Tabelle ist nicht dasselbe wie eine Codepage, in der die Codepunkte der einzelnen Zeichen angegeben werden.

Betrachten Sie das folgende Beispiel. Die ASCII-Zeichen A bis Z werden durch die Codepunkte X'41' bis X'5A' dargestellt. Um eine Sortierfolge zu beschreiben, in der diese Zeichen nacheinander (ohne Zeichen dazwischen) sortiert werden, können Sie Folgendes schreiben: X'41', X'42', ... X'59', X'5A'.

Der Hexadezimalwert eines Mehrbytezeichens wird ebenfalls als Wertigkeit verwendet. Nehmen Sie zum Beispiel an, dass die die Doppelbytezeichen A und B durch die Codepunkte X'8260' bzw. X'8261' dargestellt werden. In diesem Fall werden die Sortierwertigkeiten für X'82', X'60' und X'61' zum Sortieren dieser beiden Zeichen entsprechend ihren Codepunkten verwendet.

Die Wertigkeiten in einer Sortierfolge brauchen nicht eindeutig zu sein. Zum Beispiel könnten Sie Großbuchstaben und den entsprechenden Kleinbuchstaben jeweils dieselbe Wertigkeit zuweisen.

Die Angabe einer Sortierfolge kann einfacher sein, wenn die Sortierfolge Wertigkeiten für alle 256 Codepunkte angibt. Die Wertigkeit jedes Zeichens kann über den Codepunkt des Zeichens bestimmt werden.

In jedem Fall verwendet DB2 Universal Database™ (DB2 UDB) die Sortierfolgetabelle, die bei der Erstellung der Datenbank angegeben wurde. Wenn Sie die Mehrbytezeichen in der Reihenfolge sortieren wollen, in der sie in ihrer Codepunkttabelle auftreten, müssen Sie IDENTITY als Sortierfolge bei der Erstellung der Datenbank angeben.

Anmerkung: Für Unicode-Datenbanken werden die verschiedenen unterstützten Sortierfolgen im Abschnitt „Unicode-Implementierung in DB2 Universal Database" beschrieben.

Wenn eine Sortierfolge definiert ist, werden alle zukünftigen Zeichenvergleiche für diese Datenbank mit dieser Sortierfolge ausgeführt. Mit Ausnahme von Zeichendaten, die mit dem Attribut FOR BIT DATA oder BLOB definiert wurden, wird die Sortierfolge für alle SQL-Vergleiche und Klauseln ORDER BY verwendet, sowie für das Erstellen von Indizes und Statistiken.

In folgenden Fällen können Probleme auftreten:

- Eine Anwendung mischt sortierte Daten aus einer Datenbank mit Anwendungsdaten, die mit einer anderen Sortierfolge sortiert wurden.
- Eine Anwendung mischt sortierte Daten aus einer Datenbank mit sortierten Daten aus einer anderen Datenbank, wobei beide unterschiedliche Sortierfolgen haben.
- Eine Anwendung geht von Voraussetzungen für die sortierten Daten aus, die für die betreffende Sortierfolge nicht stimmen. Für eine bestimmte Sortierfolge kann beispielsweise gelten, dass Zahlen nach Buchstaben angeordnet werden oder nicht.

Ein letzter Punkt, der zu beachten ist, besteht darin, dass die Ergebnisse eines Sortiervorgangs, der auf einem direkten Vergleich von Zeichencodepunkten beruht, nur mit den Ergebnissen einer Abfrage übereinstimmen, die mit einer Identitätssortierfolge geordnet wurden.

Zugehörige Konzepte:

- „Character conversion" in *SQL Reference, Volume 1*
- „Unicode-Implementierung in DB2 Universal Database" auf Seite 309
- „Character comparisons based on collating sequences" in *Application Development Guide: Programming Client Applications*

Sortieren thailändischer Zeichen

Thailändisch enthält spezielle Vokale (″führende Vokale″), Tonmarkierungen und andere Sonderzeichen, die nicht sequenziell sortiert werden.

Einschränkungen:

Sie müssen entweder Ihre Datenbank mit einem thailändischen Locale und einem thailändischen codierten Zeichensatz erstellen oder eine Unicode-Datenbank erstellen.

Vorgehensweise:

Verwenden Sie beim Erstellen einer Datenbank mit Thailändisch und dem entsprechenden codierten Zeichensatz die Klausel COLLATE USING NLSCHAR im Befehl CREATE DATABASE. Wenn Sie eine Unicode-Datenbank erstellen, verwenden Sie die Klausel COLLATE USING UCA400_LTH im Befehl CREATE DATABASE.

Zugehörige Konzepte:

- „Sortierfolgen" auf Seite 303

Zugehörige Referenzen:

- „CREATE DATABASE Command" in *Command Reference*

Datums- und Zeitformate nach Gebietscodes

Die Zeichenfolgen für Datums- und Zeitformate entsprechen dem Standardformat der Werte für Datum und Uhrzeit, die dem Gebietscode der Anwendung zugeordnet sind. Dieses Standardformat kann durch Angeben der Formatoption DATETIME beim Vorkompilieren oder Binden an die Datenbank überschrieben werden.

Es folgt eine Beschreibung der Ein- und Ausgabeformate für Datum und Uhrzeit:

- Eingabezeitformat
  - Es gibt kein Standardformat für Uhrzeiteingaben.
  - Für alle Gebietscodes können alle Zeitformate eingegeben werden.
- Ausgabezeitformat
  - Der Standardwert für die Ausgabe des Zeitangaben entspricht dem lokalen Zeitformat.
- Eingabedatumsformat
  - Es gibt kein Standardformat für Datumseingaben.
  - Wenn das lokale Format für das Datum und ein ISO-, JIS-, EUR- oder USA-Datumsformat nicht übereinstimmen, wird das lokale Format für die Datumseingabe angenommen. Siehe z. B. den Eintrag für Großbritannien in Tabelle 97 auf Seite 306.
- Ausgabedatumsformat
  - Das Standardformat für das Ausgabedatum wird in Tabelle 97 auf Seite 306 gezeigt.

    Anmerkung: Tabelle 97 auf Seite 306 zeigt außerdem eine Liste der Zeichenfolgeformate für die verschiedenen Gebietscodes.

Tabelle 97. Datums- und Zeitformate nach Gebietscode

Gebietscode	Lokales Datumsformat	Lokales Zeitformat	Standardausgabedatums- format	Eingabedatumsformate
355 Albanien	jjjj-mm-tt	JIS	LOC	LOC, USA, EUR, ISO
785 Arabisch	tt/mm/jjjj	JIS	LOC	LOC, EUR, ISO
001 Australien (1)	mm-tt-jjjj	JIS	LOC	LOC, USA, EUR, ISO
061 Australien	tt-mm-jjjj	JIS	LOC	LOC, USA, EUR, ISO
032 Belgien	tt/mm/jjjj	JIS	LOC	LOC, EUR, ISO
055 Brasilien	tt.mm.jjjj	JIS	LOC	LOC, EUR, ISO
359 Bulgarien	tt.mm.jjjj	JIS	EUR	LOC, USA, EUR, ISO
001 Kanada	mm-tt-jjjj	JIS	USA	LOC, USA, EUR, ISO
002 Kanada (Französisch)	tt-mm-jjjj	ISO	ISO	LOC, USA, EUR, ISO
385 Kroatien	jjjj-mm-tt	JIS	ISO	LOC, USA, EUR, ISO
042 Tschechische Republik	jjjj-mm-tt	JIS	ISO	LOC, USA, EUR, ISO
045 Dänemark	tt-mm-jjjj	ISO	ISO	LOC, USA, EUR, ISO
358 Finnland	tt/mm/jjjj	ISO	EUR	LOC, EUR, ISO
389 FJR Republik Mazedonien	tt.mm.jjjj	JIS	EUR	LOC, USA, EUR, ISO
033 Frankreich	tt/mm/jjjj	JIS	EUR	LOC, EUR, ISO
049 Deutschland	tt/mm/jjjj	ISO	ISO	LOC, EUR, ISO
030 Griechenland	tt/mm/jjjj	JIS	LOC	LOC, EUR, ISO
036 Ungarn	jjjj-mm-tt	JIS	ISO	LOC, USA, EUR, ISO
354 Island	tt-mm-jjjj	JIS	LOC	LOC, USA, EUR, ISO
091 Indien	tt/mm/jjjj	JIS	LOC	LOC, EUR, ISO
972 Israel	tt/mm/jjjj	JIS	LOC	LOC, EUR, ISO
039 Italien	tt/mm/jjjj	JIS	LOC	LOC, EUR, ISO
081 Japan	mm/tt/jjjj	JIS	ISO	LOC, USA, EUR, ISO
082 Korea	mm/tt/jjjj	JIS	ISO	LOC, USA, EUR, ISO
001 Lateinamerika (1)	mm-tt-jjjj	JIS	LOC	LOC, USA, EUR, ISO
003 Lateinamerika	tt-mm-jjjj	JIS	LOC	LOC, EUR, ISO

Tabelle 97. Datums- und Zeitformate nach Gebietscode (Forts.)

Gebietscode	Lokales Datumsformat	Lokales Zeitformat	Standardausgabe-datums- format	Eingabe-datumsformate
031 Niederlande	tt-mm-jjjj	JIS	LOC	LOC, USA, EUR, ISO
047 Norwegen	tt/mm/jjjj	ISO	EUR	LOC, EUR, ISO
048 Polen	jjjj-mm-tt	JIS	ISO	LOC, USA, EUR, ISO
351 Portugal	tt/mm/jjjj	JIS	LOC	LOC, EUR, ISO
086 VR China	mm/tt/jjjj	JIS	ISO	LOC, USA, EUR, ISO
040 Rumänien	jjjj-mm-tt	JIS	ISO	LOC, USA, EUR, ISO
007 Russland	tt/mm/jjjj	ISO	LOC	LOC, EUR, ISO
381 Serbien/ Montenegro	jjjj-mm-tt	JIS	ISO	LOC, USA, EUR, ISO
042 Slowakei	jjjj-mm-tt	JIS	ISO	LOC, USA, EUR, ISO
386 Slowenien	jjjj-mm-tt	JIS	ISO	LOC, USA, EUR, ISO
034 Spanien	tt/mm/jjjj	JIS	LOC	LOC, EUR, ISO
046 Schweden	tt/mm/jjjj	ISO	ISO	LOC, EUR, ISO
041 Schweiz	tt/mm/jjjj	ISO	EUR	LOC, EUR, ISO
088 Taiwan	mm-tt-jjjj	JIS	ISO	LOC, USA, EUR, ISO
066 Thailand (2)	tt/mm/jjjj	JIS	LOC	LOC, EUR, ISO
090 Türkei	tt/mm/jjjj	JIS	LOC	LOC, EUR, ISO
044 Großbritannien	tt/mm/jjjj	JIS	LOC	LOC, EUR, ISO
001 USA	mm-tt-jjjj	JIS	USA	LOC, USA, EUR, ISO
084 Vietnam	tt/mm/jjjj	JIS	LOC	LOC, EUR, ISO

Anmerkungen:

1. Ländern/Regionen, die die standardmäßige Ländereinstellung C (Locale C) verwenden, wird der Gebietscode 001 zugeordnet.
2. jjjj in der buddhistischen Zeitrechnung entspricht der gregorianischen Zeitrechnung + 543 Jahre (nur Thailand).

Zugehörige Referenzen:

- „BIND Command" in *Command Reference*
- „PRECOMPILE Command" in *Command Reference*

Unicode-Zeichencodierung

Der Unicode-Zeichencodierungsstandard ist ein Codierungsschema für Zeichen mit fester Länge, das Zeichen fast aller lebenden Sprachen der Welt umfasst.

Informationen zu Unicode stehen in der aktuellsten Ausgabe des Buchs ″The Unicode Standard″ sowie auf der Website von The Unicode Consortium (www.unicode.org) zur Verfügung.

Unicode arbeitet mit zwei Codierungsformen: 8-Bitcodierung und 16-Bitcodierung. Die Standardcodierungsform ist die 16-Bitcodierung, d. h. jedes Zeichen ist 16 Bit (zwei Byte) lang und wird gewöhnlich in der Form U+hhhh angegeben, wobei hhhh der Hexadezimalcodepunkt des Zeichens ist. Obwohl die sich daraus ergebenden über 65000 Codeelemente zur Codierung der meisten Zeichen der wichtigsten Sprachen der Welt ausreichen, bietet der Unicode-Standard außerdem einen Erweiterungsmechanismus, mit dem bis zu einer Million weiterer Zeichen codiert werden können. Der Erweiterungsmechanismus arbeitet mit einem Paar aus einem hohen und einem niedrigen Ersatzzeichen zur Codierung eines erweiterten oder ergänzenden Zeichens. Das erste (oder hohe) Ersatzzeichen besitzt einen Codewert zwischen U+D800 und U+DBFF und das zweite (oder niedrige) Ersatzzeichen einen Codewert zwischen U+DC00 and U+DFFF.

UCS-2

Der Standard 10646 (ISO/IEC 10646) der International Standards Organization (ISO) und der International Electrotechnical Commission (IEC) definiert den Universalzeichensatz Universal Multiple-Octet Coded Character Set (UCS), von dem es eine 16-Bit- (2-Byte-) und eine 32-Bitversion (4-Byteversion) gibt. UCS-2 ist mit der 16-Bit-Unicode-Form ohne Ersatzzeichen identisch. UCS-2 kann sämtliche Zeichen (16-Bitzeichen) codieren, die im Repertoire der Unicode Version 3.0 definiert sind. Zwei UCS-2-Zeichen (ein hohes Ersatzzeichen, gefolgt von einem niedrigen Ersatzzeichen) sind erforderlich, um jedes der neuen ergänzenden Zeichen zu codieren, die mit Unicode-Version 3.1 eingeführt wurden. Diese ergänzenden Zeichen wurden außerhalb der ursprünglichen mehrsprachigen 16-Bit-Basisebene (Basic Multilingual Plane - BMP oder Plane 0) definiert.

UTF-8

16-Bit-Unicode-Zeichen stellen für byteorientierte und ASCII-basierte Anwendungen und Dateisysteme ein großes Problem dar. Zum Beispiel könnten Anwendungen, die nicht auf Unicode ausgerichtet sind, die führenden acht Nullbit des Großbuchstabenzeichens 'A' (U+0041) als Einzelbytezeichen NULL fehlinterpretieren.

UTF-8 (UCS Transformation Format 8) ist eine algorithmische Umsetzung, mit der Unicode-Zeichen mit fester Länge in Bytefolgen variabler Länge umgesetzt werden. In UTF-8 werden ASCII-Zeichen und Steuerzeichen durch ihren gewöhnlichen Einzelbytecode dargestellt. Andere Zeichen werden jedoch zwei oder mehr Byte lang. In UTF-8 können sowohl nicht ergänzende als auch ergänzende Zeichen codiert werden.

UTF-16

ISO/IEC 10646 definiert außerdem eine Erweiterungstechnik zum Codieren einiger UCS-4-Zeichen mit Hilfe von zwei UCS-2-Zeichen. Diese Erweiterung wird als UTF-16 bezeichnet und entspricht der 16-Bit-Unicode-Form mit Ersatzzeichen. Insgesamt besteht das UTF-16-Zeichenrepertoire aus allen UCS-2-Zeichen und der zusätzlichen eine Million Zeichen, auf die über die Ersatzzeichenpaare zugegriffen werden kann.

Bei der Serialisierung von 16-Bit-Unicode-Zeichen in Byte platzieren einige Prozessoren das höchstwertige Byte in Anfangsposition (Big-Endian-Reihenfolge),

während andere Prozessoren das niedrigstwertige Byte zuerst platzieren (Little-Endian-Reihenfolge). Die Standard-Bytereihenfolge für Unicode ist die Big-Endian-Reihenfolge.

Die Anzahl von Byte für jedes UTF-16-Zeichen im UTF-8-Format ist Tabelle 98 zu entnehmen.

Tabelle 98. UTF-8-Bitverteilung

Codewert (binär)	UTF-16 (binär)	1. Byte (binär)	2. Byte (binär)	3. Byte (binär)	4. Byte (binär)
00000000 0xxxxxxx	00000000 0xxxxxxx	0xxxxxxx			
00000yyy yyxxxxxx	00000yyy yyxxxxxx	110yyyyy	10xxxxxx		
zzzzyyyy yyxxxxxx	zzzzyyyy yyxxxxxx	1110zzzz	10yyyyyy	10xxxxxx	
uuuuu zzzzyyyy yyxxxxxx	110110ww wwzzzzyy 110111yy yyxxxxxx	11110uuu (mit uuuuu = wwww+1)	10uuzzzz	10yyyyyy	10xxxxxx

In den oben gezeigten Angaben entspricht eine Reihe von u, w, x, y und z jeweils der Bitdarstellung des Zeichens. Zum Beispiel wird U+0080 binär in 11000010 10000000 umgesetzt, und das Ersatzzeichenpaar U+D800 U+DC00 wird binär in 11110000 10010000 10000000 10000000 umgesetzt.

Zugehörige Konzepte:

- „Unicode-Implementierung in DB2 Universal Database" auf Seite 309
- „Unicode-Behandlung der Datentypen" auf Seite 312
- „Unicode-Literale" auf Seite 314

Zugehörige Tasks:

- „Erstellen einer Unicode-Datenbank" auf Seite 314

Unicode-Implementierung in DB2 Universal Database

DB2® Universal Database (DB2 UDB) unterstützt UTF-8 und UCS-2.

Wenn eine Unicode-Datenbank erstellt wird, werden CHAR-, VARCHAR-, LONG VARCHAR- und CLOB-Daten im UTF-8-Format und GRAPHIC-, VARGRAPHIC-, LONG VARGRAPHIC- und DBCLOB-Daten im UCS-2-Format (Big-Endian) gespeichert.

In den Versionen von DB2 UDB vor Version 7.2 FixPak 4 behandelt DB2 UDB die beiden Zeichen in einem Ersatzzeichenpaar als zwei unabhängige Unicode-Zeichen. Daher führt die Umsetzung des Zeichenpaares von UTF-16/UCS-2 in UTF-8 zu zwei 3-Bytefolgen. Seit DB2 UDB Version 7.2 FixPak 4 erkennt DB2 UDB

Ersatzzeichenpaare bei der Umsetzung zwischen UTF-16/UCS-2 und UTF-8, so dass ein Paar aus UTF-16-Ersatzzeichen in eine 4-Byte-UTF-8-Zeichenfolge umgesetzt wird. In anderen Verwendungen behandelt DB2 UDB ein Ersatzzeichenpaar weiterhin wie zwei unabhängige UCS-2-Zeichen. Sie können Ergänzungszeichen in DB2 UDB-Unicode-Datenbanken sicher speichern, sofern Sie wissen, wie diese von den anderen Zeichen (Nichtergänzungszeichen) zu unterscheiden sind.

DB2 UDB behandelt jedes Unicode-Zeichen, einschließlich dieser Zeichen (außer Leerzeichen), wie das mit dem Akzentzeichen Akut kombinierte Zeichen (U+0301, KOMBINIERENDER AKZENT AKUT), als einzelnes Zeichen. Aus diesem Grund würde DB2 UDB nicht erkennen, dass das Zeichen LATIN KLEINER BUCHSTABE A MIT AKUT (U+00E1) kanonisch äquivalent zum Zeichen LATIN KLEINER BUCHSTABE A (U+0061), gefolgt vom Zeichen KOMBINIERENDER AKUT (U+0301), ist.

Die Standardsortierfolge für eine UCS-2-Unicode-Datenbank ist IDENTITY, die die Zeichen nach ihren Codepunkten anordnet. Daher werden standardmäßig alle Unicode-Zeichen nach ihren Codepunkten geordnet und verglichen. Für die nicht ergänzenden Unicode-Zeichen stimmen die binären Sortierfolgen bei der Codierung in UTF-8 und UCS-2 überein. Wenn Sie jedoch ein Ergänzungszeichen haben, für dessen Codierung ein Ersatzzeichenpaar erforderlich ist, wird das Zeichen in der UTF-8-Codierung an das Ende sortiert, während das gleiche Zeichen in UCS-2-Codierung irgendwo in der Mitte eingereiht wird, wobei seine beiden Ersatzzeichen getrennt werden können. Der Grund hierfür besteht darin, dass das erweiterte Zeichen bei der Codierung in UTF-8 einen 4-Bytewert der Form 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx besitzt, der größer ist als die UTF-8-Codierung von U+FFFF, nämlich X'EFBFBF'. In der UCS-2-Codierung wird das gleiche Ergänzungszeichen hingegen als Paar aus einem hohen und einem niedrigen UCS-2-Ersatzzeichen codiert und besitzt die Binärform 1101 1000 xxxx xxxx 1101 1100 xxxx xxxx, die jedoch einen kleineren Wert als die UCS-2-Codierung von U+FFFF darstellt.

Eine Unicode-Datenbank kann außerdem mit der Sortierfolgenoption IDENTITY_16BIT erstellt werden. Die IDENTITY_16BIT-Sortierung implementiert den CESU-8-Algorithmus (*Compatibility Encoding Scheme for UTF-16: 8-Bit*), wie er im Unicode Technical Report #26 spezifiziert wird, der auf der Website des Unicode Technical Consortium (www.unicode.org) zur Verfügung steht. CESU-8 ist, abgesehen von den Unicode-Ergänzungszeichen, binär mit UTF-8 identisch. Die Unicode-Ergänzungszeichen sind eben die Zeichen, die außerhalb der mehrsprachigen 16-Bit-Basisebene (Basic Multilingual Plane - BMP oder Plane 0) definiert sind. In der UTF-8-Codierung wird ein Ergänzungszeichen durch eine 4-Bytefolge dargestellt, wohingegen das gleiche Zeichen in CESU-8 zwei 3-Bytefolgen erfordert. Durch die Verwendung der Sortierfolgenoption IDENTITY_16BIT ergibt sich die gleiche Sortierreihenfolge für Zeichen- und Grafikdatentypen.

DB2 UDB Version 8.2 unterstützt zwei neue Schlüsselwörter für die Sortierfolge für Unicode-Datenbanken: UCA400_NO und UCA400_LTH. Die UCA400_NO-Sortierung implementiert den UCA-Algorithmus (Unicode Collation Algorithm) auf der Basis von Unicode Standard Version 4.00 mit implizit aktivierter Normalisierung. Die UCA400_LTH-Sortierung implementiert ebenfalls den UCA-Algorithmus Version 4.00, sortiert jedoch thailändische Zeichen in der im Royal Thai Dictionary festgelegten Reihenfolge. Details des UCA-Algorithmus finden Sie im Unicode Technical Standard #10, der auf der Website des Unicode Consortium (www.unicode.org) zur Verfügung steht.

Alle länderspezifischen Parameter wie Datums- oder Uhrzeitformat, Dezimaltrennzeichen u. a. basieren auf dem aktuellen Gebiet des Clients.

Eine Unicode-Datenbank erlaubt Verbindungen von jeder Codepage, die von DB2 UDB unterstützt wird. Der Datenbankmanager führt die Codepagekonvertierung automatisch für Zeichen- und Grafikzeichenfolgen zwischen der Codepage des Clients und Unicode aus.

Jeder Client ist durch die Menge von Zeichen, die Eingabemethode und die von seiner Umgebung unterstützten Schriftarten beschränkt. Die UCS-2-Datenbank selbst jedoch akzeptiert und speichert alle UCS-2-Zeichen. Jeder Client arbeitet also mit einer Untergruppe von UCS-2-Zeichen, der Datenbankmanager lässt jedoch die Gesamtheit von UCS-2-Zeichen zu.

Wenn Zeichen von einer lokalen Codepage in Unicode umgesetzt werden, wird möglicherweise die Byteanzahl erweitert. Vor Version 8 wurden Zeichendaten je nach der Semantik von SQL-Anweisungen möglicherweise als in der Codepage des Clients codiert markiert und der Datenbankserver verarbeitete die gesamte Anweisung in der Codepage des Clients. Diese Art der Verarbeitung hätte zu einer potenziellen Erweiterung der Daten führen können. Ab Version 8 arbeitet der Datenbankserver, sobald er eine SQL-Anweisung empfangen hat, nur in der Codepage des Datenbankservers. In diesem Fall kommt es zu keiner Änderung der Größe.

Nummern von Codepages/IDs für codierten Zeichensatz

Bei IBM® wurde die UCS-2-Codepage als Codepage 1200 mit wachsendem Zeichensatz registriert. Das heißt, dass sich durch das Hinzufügen neuer Zeichen zu einer Codepage die Nummer der Codepage nicht ändert. Codepage 1200 verweist immer auf die aktuelle Version von Unicode.

Eine spezifische Version des UCS-Standards, wie durch Unicode 2.0 und ISO/IEC 10646-1 definiert, wurde ebenfalls bei IBM als ID für codierten Zeichensatz (CCSID) 13488 registriert. Diese ID für codierten Zeichensatz wird intern von DB2 UDB zum Speichern von Grafikzeichenfolgen in IBM eucJP (Japan)- und IBM eucTW (Taiwan)-Datenbanken verwendet. Die ID für codierten Zeichensatz 13488 und die Codepage 1200 verweisen beide auf UCS-2 und werden auf dieselbe Weise behandelt mit Ausnahme des Werts ihres „Doppelbyte"-Leerzeichens (DBCS):

CP/CCSID	Einzelbyte-Leerzeichen (SBCS)	Doppelbyte-Leerzeichen (DBCS)
1200	n/v	U+0020
13488	n/v	U+3000

Anmerkung: In einer UCS-2-Datenbank hat U+3000 keine spezielle Bedeutung.

Bei Konvertierungstabellen werden die gleichen Tabellen für beide verwendet, da Codepage 1200 eine Obermenge von CCSID 13488 ist.

Bei IBM wurde UTF-8 als ID für codierten Zeichensatz 1208 mit wachsendem Zeichensatz registriert (manchmal auch als Codepage 1208 bezeichnet). Wenn neue Zeichen zum Standard hinzugefügt werden, ändert sich diese Nummer (1208) nicht.

Die MBCS-Codepage-Nummer ist 1208. Dies ist die Codepage-Nummer der Datenbank und die Codepage von Zeichenfolgedaten in der Datenbank. Die Doppelbyte-Codepage-Nummer für UCS-2 ist 1200. Dies ist die Codepage von Grafikzeichenfolgedaten in der Datenbank.

Unterschiede zwischen den Sortieralgorithmen für Thailändisch und Unicode

Der Sortieralgorithmus, der in einer thailändischen Datenbank mit Thai Industrial Standard (TIS) TIS620-1 (Codepage 874) mit der Sortierfolgeoption NLSCHAR verwendet wird, ist dem Sortieralgorithmus einer Unicode-Datenbank mit UCA400_LTH-Sortierfolge ähnlich, jedoch nicht mit ihm identisch.
Es bestehen folgende Unterschiede:

- Bei der Sortierung von TIS620-1-Daten besitzt jedes Zeichen nur eine Wertigkeit, und diese Wertigkeit wird bei der Sortierfolgebestimmung zum Vergleich mit der Wertigkeit eines anderen Zeichens verwendet. Bei der Sortierung von Unicode-Daten besitzt jedes Zeichen mehrere Wertigkeiten, und alle Wertigkeiten eines Zeichen können bei der Sortierfolgebestimmung verwendet werden.
- Bei der Sortierung von TIS620-1-Daten besitzen das Leerzeichen X'20', der Silbentrennungsstrich X'2D' und das Punktzeichen X'2E' sämtlich kleinere Wertigkeiten als alle thailändischen Zeichen. Bei der Sortierung von Unicode-Daten werden diese drei Zeichen hingegen als Interpunktionszeichen betrachtet und nur dann zum Vergleich herangezogen, wenn alle anderen Zeichen in zwei zu vergleichenden Zeichenfolgen gleich sind.
- Das Paiyannoi-Zeichen X'CF' und das Maiyamok-Zeichen X'E6' in einer TIS620-1-Datenbank werden als Interpunktionszeichen behandelt, wenn sie auf andere thailändische Zeichen folgen, und als normale Zeichen mit eigenen Wertigkeiten, wenn sie am Anfang einer Zeichenfolge auftreten. Dieselben beiden Zeichen in einer Unicode-Datenbank (U+0E2F bzw. U+0E46) werden stets als Interpunktionszeichen behandelt und nur zum Vergleich herangezogen, wenn alle anderen Zeichen in zwei zu vergleichenden Zeichenfolgen gleich sind.

Weitere Informationen zu thailändischen Zeichen finden Sie in Kapitel 10.1 des Buchs "Thai of the Unicode Standard", Version 4.0, ISBN 0-321-18578-1.

Zugehörige Konzepte:

- „Unicode-Zeichencodierung" auf Seite 307
- „Unicode-Behandlung der Datentypen" auf Seite 312
- „Unicode-Literale" auf Seite 314

Zugehörige Tasks:

- „Erstellen einer Unicode-Datenbank" auf Seite 314

Unicode-Behandlung der Datentypen

Alle von DB2® Universal Database (DB2 UDB) unterstützten Datentypen werden auch in einer UCS-2-Datenbank unterstützt. Insbesondere Grafikzeichenfolgedaten werden für eine UCS-2-Datenbank unterstützt und in UCS-2/Unicode gespeichert. Jeder Client, einschließlich Clients mit SBCS-Zeichensätzen, kann mit Grafikzeichenfolgedatentypen in UCS-2/Unicode arbeiten, wenn er mit einer UCS-2-Datenbank verbunden ist.

Eine UCS-2-Datenbank verhält sich wie jede beliebige MBCS-Datenbank, bei der die Zeichenfolgedaten in der Anzahl von Byte gemessen werden. Beim Arbeiten

mit Zeichenfolgedaten in UTF-8 sollte nicht davon ausgegangen werden, dass jedes Zeichen ein Byte lang ist. Bei UTF-8-Mehrbytecodierung ist jedes ASCII-Zeichen ein Byte lang, andere Zeichen nehmen jedoch jeweils zwei bis vier Byte ein. Dies sollte beim Definieren von CHAR-Feldern berücksichtigt werden. Je nach Verhältnis von ASCII-Zeichen zu anderen Zeichen kann ein CHAR-Feld der Größe von *n* Byte zwischen *n*/4 bis *n* Zeichen enthalten.

Die Verwendung der UTF-8-Zeichenfolgecodierung statt des UCS-2-Grafikzeichenfolgedatentyps wirkt sich ebenfalls auf die Gesamtspeicheranforderungen aus. Wenn die Mehrheit der Zeichen ASCII-Zeichen sind und einige andere Zeichen dazwischenstehen, kann das Speichern von UTF-8-Daten die günstigere Alternative sein, da der Speicherbedarf eher einem Byte pro Zeichen entspricht. Wenn dagegen die Mehrheit der Zeichen keine ASCII-Zeichen sind, die auf drei oder vier Byte lange UTF-8-Folgen erweitert werden (z. B. Ideogramme), ist das UCS-2-Grafikzeichenfolgeformat möglicherweise eine bessere Alternative, da jede drei Byte lange UCS-2-Zeichenfolge zu einem 16-Bit-UCS-2-Zeichen wird, während jede vier Byte lange UTF-8-Zeichenfolge zu zwei 16-Bit-UCS-2-Zeichen wird.

In MBCS-Umgebungen operieren SQL-Funktionen, die mit Zeichenfolgen arbeiten, z. B. LENGTH, SUBSTR, POSSTR, MAX, MIN u. ä., mit der Anzahl von „Byte" statt mit der Anzahl von „Zeichen". Diese Funktionsweise ist in einer UCS-2-Datenbank unverändert. Sie sollten jedoch besonders vorsichtig sein, wenn Sie relative Positionen und Längen für eine UCS-2-Datenbank angeben, da diese Werte immer im Kontext der Datenbank-Codepage definiert werden. Im Fall einer UCS-2-Datenbank sollten diese relativen Positionen daher in UTF-8 definiert werden. Da einige Einzelbytezeichen in UTF-8 mehr als ein Byte benötigen, sind SUBSTR-Indizes, die für eine Einzelbytedatenbank gültig sind, möglicherweise für eine UCS-2-Datenbank nicht gültig. Wenn Sie falsche Indizes angeben, wird SQLCODE-Wert -191 (SQLSTATE 22504) zurückgegeben.

SQL-Datentypen CHAR werden in Benutzerprogrammen vom Datentyp `char` (in der Programmiersprache C) unterstützt. SQL-Datentypen GRAPHIC werden von `sqldbchar` in Benutzerprogrammen unterstützt. Bei einer UCS-2-Datenbank sind `sqldbchar`-Daten immer im Big-Endian-Format (höherwertiges Byte zuerst). Wenn ein Anwendungsprogramm mit einer UCS-2-Datenbank verbunden ist, werden Zeichenfolgedaten zwischen der Codepage der Anwendung und UTF-8 und Grafikzeichenfolgedaten zwischen der Anwendungsgrafikcodepage und UCS-2 durch DB2 UDB umgesetzt.

Wenn Daten von einer Unicode-Datenbank zu einer Anwendung abgerufen werden, die keine SBCS, EUC oder Unicode-Codepage verwendet, wird das definierte Substitutionszeichen für jedes aufgefüllte Leerzeichen in einer grafischen Spalte zurückgegeben. DB2 UDB füllt grafische Unicode-Spalten mit fester Länge mit ASCII-Leerzeichen (U+0200) auf, einem Zeichen, das kein Äquivalent in reinen DBCS-Codepages besitzt. Daher wird jedes ASCII-Leerzeichen, das beim beim Auffüllen der grafischen Spalte verwendet wird, bei der Abfrage in das Substitutionszeichen konvertiert. Entsprechend wird in einer DATE, TIME oder TIMESTAMP-Zeichenfolge jedes SBCS-Zeichen, das kein reines DBCS-Äquivalent besitzt, in das Substitutionszeichen konvertiert, wenn es von einer Unicode-Datenbank in eine Anwendung abgerufen wird, die keine SBCS, EUC oder Unicode-Codepage verwendet.

Anmerkung: Bis zur Version 8 wurden Grafikzeichenfolgedaten immer als UCS-2 angenommen. Um die Abwärtskompatibilität zu Anwendungen zu gewährleisten, die vom vorherigen Verhalten von DB2 UDB abhängen, wurde die Registrierdatenbankvariable

DB2GRAPHICUNICODESERVER eingeführt. Ihr Standardwert ist OFF. Das Ändern des Werts dieser Variable in ON bewirkt, dass sich DB2 UDB wie in früheren Versionen verhält und annimmt, dass die Grafikzeichenfolgedaten immer im UCS-2-Format vorliegen. Zusätzlich prüft der DB2 UDB-Server die Version von DB2 UDB, die auf dem Client ausgeführt wird, und simuliert das Verhalten von Version 7, wenn auf dem Client Version 7 ausgeführt wird.

Zugehörige Konzepte:

- „Unicode-Zeichencodierung" auf Seite 307
- „Unicode-Implementierung in DB2 Universal Database" auf Seite 309

Erstellen einer Unicode-Datenbank

Vorgehensweise:

Standardmäßig werden Datenbanken in der Codepage der Anwendung erstellt, die sie erstellt. Wenn Sie daher Ihre Datenbank von einem Unicode-Client (UTF-8) aus erstellen (z. B. Locale UNIVERSAL von AIX) oder wenn die Registriervariable DB2CODEPAGE auf dem Client auf 1208 gesetzt ist, wird Ihre Datenbank als Unicode-Datenbank erstellt. Alternativ dazu können Sie explizit "UTF-8" als Namen für einen codierten Zeichensatz (CODESET) angeben und jeden gültigen Gebietscode (TERRITORY) verwenden, der von DB2 Universal Database™ (DB2 UDB) unterstützt wird.

Führen Sie zum Beispiel den folgenden Befehl aus, um eine Unicode-Datenbank mit dem Gebietscode für die Vereinigten Staaten von Amerika zu erstellen:

```
DB2 CREATE DATABASE dbname USING CODESET UTF-8 TERRITORY US
```

Zum Erstellen einer Unicode-Datenbank mit der API **sqlecrea** sollten Sie die Werte in *sqledbterritoryinfo* entsprechend definieren. Setzen Sie z. B. SQLDBCODESET auf den Wert `UTF-8` und SQLDBLOCALE auf einen gültigen Gebietscode (z. B. `US`).

Zugehörige Konzepte:

- „Unicode-Implementierung in DB2 Universal Database" auf Seite 309

Zugehörige Referenzen:

- „sqlecrea - Create Database" in *Administrative API Reference*
- „CREATE DATABASE Command" in *Command Reference*

Unicode-Literale

Unicode-Literale können auf zwei Arten angegeben werden:

- Als Grafikzeichenfolgekonstante mit dem Format G'...' oder N'....' : Jedes auf diese Art angegebene Literal wird vom Datenbankmanager von der Codepage der Anwendung in 16-Bit-Unicode umgesetzt.
- Als hexadezimale Unicode-Zeichenfolge mit dem Format UX'....' oder GX'....': Die in Anführungszeichen nach UX oder GX angegebene Konstante muss ein Mehrfaches von vier hexadezimalen Ziffern in Big-Endian-Reihenfolge sein. Jede Gruppe aus vier Ziffern stellt jeweils einen 16-Bit-Unicode-Codepunkt dar. Beachten Sie, dass Ersatzzeichen immer in Paaren erscheinen, sodass Sie zur Darstellung des hohen und niedrigen Ersatzzeichens zwei vierziffrige Gruppen benötigen.

Bei Verwendung des Befehlszeilenprozessors (CLP) ist die erste Methode einfacher, wenn das UCS-2-Zeichen in der Codepage der lokalen Anwendung vorhanden ist (z. B. zur Eingabe von Zeichen der Codepage 850 von einem Terminal, das Codepage 850 verwendet). Die zweite Methode sollte für Zeichen verwendet werden, die außerhalb des Umfangs der Codepage der Anwendung liegen (z. B. zum Angeben von japanischen Zeichen von einem Terminal, das Codepage 850 verwendet).

Zugehörige Konzepte:

- „Unicode-Zeichencodierung" auf Seite 307
- „Unicode-Implementierung in DB2 Universal Database" auf Seite 309

Zugehörige Referenzen:

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

Zeichenfolgevergleiche in einer Unicode-Datenbank

Mustererkennung ist ein Bereich, in dem sich vorhandene MBCS-Datenbanken leicht von einer UCS-2-Datenbank unterscheiden.

MBCS-Datenbanken in DB2® Universal Database (DB2 UDB) verhalten sich wie folgt: Wenn der Übereinstimmungsausdruck MBCS-Daten enthält, kann das Muster sowohl SBCS- als auch Nicht-SBCS-Zeichen enthalten. Die Sonderzeichen im Muster werden folgendermaßen interpretiert:

- Ein SBCS-Unterstreichungszeichen halber Länge steht für ein SBCS-Zeichen.
- Ein Nicht-SBCS-Unterstreichungszeichen voller Länge steht für ein Nicht-SBCS-Zeichen.
- Ein Prozentzeichen (entweder SBCS-Zeichen halber Länge oder Nicht-SBCS-Zeichen voller Länge) steht für null oder mehr SBCS- oder Nicht-SBCS-Zeichen.

In einer Unicode-Datenbank gibt es keine echte Unterscheidung zwischen „Einzelbytezeichen" und „Doppelbytezeichen". Obwohl das UTF-8-Format eine „Mischbytecodierung" von Unicode-Zeichen ist, gibt es in UTF-8 keine echte Unterscheidung zwischen SBCS- und Nicht-SBCS-Zeichen. Jedes Zeichen ist ein Unicode-Zeichen, unabhängig von der Anzahl der Byte im UTF-8-Format. In einer Unicode-Spalte des Typs GRAPHIC ist jedes Nichtergänzungszeichen, einschließlich des Unterstreichungszeichens halber Länge (U+005F) und des Prozentzeichens halber Länge (U+0025), zwei Byte lang. Bei Unicode-Datenbanken werden die Sonderzeichen in den Mustern wie folgt interpretiert:

- Bei Zeichenfolgen steht ein Unterstreichungszeichen halber Länge (X'5F') bzw. ein Unterstreichungszeichen voller Länge (X'EFBCBF') für nur ein Unicode-Zeichen. Ein Prozentzeichen halber Länge (X'25') bzw. ein Prozentzeichen voller Länge (X'EFBC85') steht für null oder mehr Unicode-Zeichen.
- Bei Grafikzeichenfolgen steht ein Unterstreichungszeichen halber Länge (U+005F) bzw. ein Unterstreichungszeichen voller Länge (U+FF3F) für nur ein Unicode-Zeichen. Ein Prozentzeichen halber Länge (U+0025) bzw. ein Prozentzeichen voller Länge (U+FF05) steht für null oder mehr Unicode-Zeichen.

Anmerkung: Sie benötigen zwei Unterstreichungszeichen, um eine Entsprechung für ein ergänzendes Unicode-Grafikzeichen anzugeben, da ein solches Zeichen durch zwei UCS-2-Zeichen in einer GRAPHIC-Spalte

dargestellt wird. Nur ein Unterstreichungszeichen wird benötigt, um ein Unicode-Ergänzungszeichen in einer CHAR-Spalte zu bezeichnen.

Der optionale „Escapeausdruck", der ein Zeichen angibt, das verwendet wird, um die spezielle Bedeutung des Unterstreichungszeichens und des Prozentzeichens aufzuheben, kann durch eines der folgenden Elemente angegeben werden:

- Eine Konstante
- Ein Sonderregister
- Eine Hostvariable
- Eine Skalarfunktion, deren Operanden beliebige der obigen Elemente sind
- Ein Ausdruck, der beliebige der obigen Elemente verkettet

Dabei gelten folgende Einschränkungen:

- Kein Element in dem Ausdruck darf den Datentyp LONG VARCHAR, CLOB, LONG VARGRAPHIC oder DBCLOB haben. Außerdem darf keine Variable enthalten sein, die auf eine BLOB-Datei verweist.
- Bei CHAR-Spalten muss das Ergebnis des Ausdrucks ein Zeichen oder eine Binärzeichenfolge sein, die genau ein (1) Byte (SQLSTATE 22019) enthält. Bei GRAPHIC-Spalten muss das Ergebnis des Ausdrucks ein Zeichen (SQLSTATE 22019) sein.

Zugehörige Konzepte:

- „Unicode-Zeichencodierung" auf Seite 307
- „Unicode-Implementierung in DB2 Universal Database" auf Seite 309

Zugehörige Referenzen:

- „Character strings" in *SQL Reference, Volume 1*
- „Graphic strings" in *SQL Reference, Volume 1*

Installieren der früheren Tabellen zur Konvertierung zwischen Codepage 1394 und Unicode

Die Konvertierungstabellen für Codepage 1394 (die auch als Shift JIS X0213 bezeichnet wird) und Unicode wurden erweitert. Die Konvertierung zwischen der japanischen Codepage Shift JIS X0213 (1394) und Unicode entspricht nun dem letzten Standard ISO/IEC 10646-1:2000 Amendment-1 für JIS-X0213-Zeichen. Die vorige Version der Konvertierungstabellen ist über FTP unter `ftp://ftp.software.ibm.com/ps/products/db2/info/vr8/conv/` verfügbar.

Vorgehensweise:

Gehen Sie wie folgt vor, um die früheren Definitionen zur Konvertierung zwischen Shift JIS X0213 und Unicode zu installieren:

1. Stoppen Sie das DB2 Universal Database™-Exemplar (DB2 UDB).
2. Geben Sie in Ihren Webbrowser die Adresse `ftp://ftp.software.ibm.com/ps/products/db2/info/vr8/conv/` ein oder stellen Sie eine Verbindung mit der Site `ftp.software.ibm.com` über FTP her. Dieser FTP-Server bietet einen anonymen Zugang.
3. Wenn Sie die Verbindung über die Befehlszeile herstellen, melden Sie sich an, indem Sie `anonymous` als Benutzer-ID und Ihre E-Mail-Adresse als Kennwort angeben.

4. Wechseln Sie nach erfolgter Anmeldung in das Verzeichnis für Konvertierungstabellen:

   ```
   cd ps/products/db2/info/vr8/conv
   ```

5. Kopieren Sie die beiden Dateien 1394ucs4.cnv und ucs41394.cnv binär in Ihr Verzeichnis `sqllib/conv/`.
6. Starten Sie das DB2 UDB-Exemplar erneut.

Zugehörige Konzepte:

• „Unicode-Implementierung in DB2 Universal Database" auf Seite 309

Zugehörige Referenzen:

• „Unterstützte Gebietscodes und Codepages" auf Seite 265

Alternative Unicode-Konvertierungstabellen für ID für codierten Zeichensatz (CCSID) 943

Einige Zeichen im codierten Zeichensatz mit der ID 943 (CCSID 943) haben jeweils zwei Codepunkte. Einer der Codepunkte wird gemeinhin als NEC-Codepunkt bezeichnet. Der andere ist allgemein als IBM® Codepunkt bekannt. Andere Zeichen in CCSID 943 besitzen ebenfalls zwei Codepunkte, das heißt einen NEC-Codepunkt und einen JIS-Codepunkt. Zum Beispiel kann die römische Ziffer 1 sowohl als X'8754' (NEC-Codepunkt) als auch als X'FA4A' (IBM Codepunkt) codiert werden. Ganz ähnlich kann das mathematische Symbol für den Vereinigungsmengenoperator durch die beiden Codepunkte X'879C' (NEC-Codepunkt) und X'81BE' (JIS-Codepunkt) dargestellt werden.

Wenn die Standardkonvertierungstabellen von DB2® Universal Database (DB2 UDB) zur Konvertierung von CCSID 943 in Unicode verwendet werden und ein Symbol durch zwei Codepunkte dargestellt werden kann, werden beide Codepunkte in das gleiche Unicode-Zeichen konvertiert. Bei der Konvertierung von Unicode in CCSID 943 werden die Zeichen nur in die IBM oder JIS-Codepunkte konvertiert. Wenn Sie zum Beispiel die Standardkonvertierungstabellen zur Konvertierung der römischen Ziffer 1 von CCSID 943 in Unicode und wieder zurück verwenden, geschieht Folgendes:

- X'FA4A' (der IBM Codepunkt) wird in U+2160 und anschließend zurück in X'FA4A' konvertiert.
- X'8754' (der NEC-Codepunkt) wird jedoch ebenfalls in U+2160 und anschließend zurück in X'FA4A' konvertiert.

Wenn die gleiche Konvertierung mit der Microsoft®-Version der Unicode-Konvertierungstabellen für CCSID 943 ausgeführt wird, werden die Zeichen X'FA4A' und X'8754' analog in das Zeichen X'8754' zurückkonvertiert.

Darüber hinaus gibt es in CCSID 943 Zeichen, die abhängig von den verwendeten Konvertierungstabellen in verschiedene Unicode-Zeichen konvertiert werden können. Zum Beispiel wird das CCSID 943-Zeichen X'815C' mit den DB2 UDB-Standardkonvertierungstabellen in das Unicode-Zeichen U+2014, mit den Microsoft-Konvertierungstabellen jedoch in das Zeichen U+2015 konvertiert. Wenn Sie die Microsoft-Version der Konvertierungstabellen mit DB2 UDB verwenden wollen, müssen Sie die DB2 UDB-Konvertierungstabellen durch die Microsoft-Konvertierungstabellen ersetzen.

Die Verwendung dieser Microsoft-Konvertierungstabellen ist auf geschlossene Umgebungen zwischen einer DB2 UDB-Datenbank mit CCSID 943 und DB2 UDB-Clients mit CCSID 943 beschränkt, in denen sowohl die Clients als auch die Datenbank mit der Microsoft-Version der Konvertierungstabellen arbeiten. Wenn Sie einen DB2 UDB-Client mit den DB2 UDB-Standardkonvertierungstabellen und einen anderen DB2 UDB-Client mit der Microsoft-Version der Konvertierungstabellen haben und beide Clients mit der gleichen DB2 UDB-Datenbank mit CCSID 943 verbunden sind, ist es möglich, dass das gleiche Zeichen mit zwei verschiedenen Codepunkten in der Datenbank gespeichert wird.

Zugehörige Konzepte:

- „Unicode-Zeichencodierung" auf Seite 307

Zugehörige Tasks:

- „Ersetzen der Unicode-Konvertierungstabellen für CCSID 943 durch die Microsoft-Konvertierungstabellen" auf Seite 318

Ersetzen der Unicode-Konvertierungstabellen für CCSID 943 durch die Microsoft-Konvertierungstabellen

Zur Konvertierung zwischen dem codierten Zeichensatz mit der ID 943 (CCSID 943) und Unicode werden die Standardkonvertierungstabellen von DB2 Universal Database™ (DB2 UDB) verwendet. Wenn Sie eine andere Version der Konvertierungstabellen, zum Beispiel die Microsoft-Version, verwenden wollen, müssen Sie die Standarddateien mit den Konvertierungstabellen (.cnv-Dateien) ersetzen.

Voraussetzungen:

Bevor Sie die vorhandenen Konvertierungstabellendateien für Codepages im Verzeichnis `sqllib/conv` ersetzen, sollten Sie die Dateien für den Fall sichern, dass Sie die Ersetzung rückgängig machen wollen. Unter UNIX ist das Verzeichnis `sqllib/conv` mit dem Installationspfad von DB2 UDB durch eine Programmverbindung (Link) verbunden.

Einschränkungen:

Damit diese Methode wirksam ist, müssen die Konvertierungstabellen aller DB2 UDB-Clients, die eine Verbindung zur gleichen Datenbank herstellen, geändert werden. Anderenfalls könnten verschiedene Clients das gleiche Zeichen mit unterschiedlichen Codepunkten speichern.

Vorgehensweise:

Gehen Sie wie folgt vor, um die DB2 UDB-Standardkonvertierungstabellen zur Konvertierung zwischen CCSID 943 und Unicode zu ersetzen:

1. Kopieren Sie `sqllib/conv/ms/0943ucs2.cnv` nach `sqllib/conv/0943ucs2.cnv`.
2. Kopieren Sie `sqllib/conv/ms/ucs20943.cnv` nach `sqllib/conv/ucs20943.cnv`.
3. Starten Sie DB2 UDB erneut.

Zugehörige Konzepte:

- „Alternative Unicode-Konvertierungstabellen für ID für codierten Zeichensatz (CCSID) 943" auf Seite 317

Anhang C. Aktivieren der Unterstützung großer Seiten in einer 64-Bit-Umgebung (AIX)

Zusätzlich zur traditionellen Seitengröße von 4 KB unterstützt der POWER4-Prozessor in den IBM eServer pSeries-Systemen eine neue Seitengröße von 16 MB. AIX 5L für POWER Version 5.1 mit Recommended Maintenance Package 5100-02 bzw. Version 5.2 enthält Unterstützung für Seiten einer Größe von 16 MB. Beim Einsatz in einer solchen Umgebung kann DB2 Universal Database™ (DB2 UDB) für AIX 64-Bit Edition für die Verwendung dieser großen Seiten aktiviert werden.

Die Verwendung großer Seiten ist primär für eine Leistungsverbesserung hochleistungsfähiger Datenverarbeitungsanwendungen gedacht. Anwendungen, die intensiven Zugriff auf den Hauptspeicher erfordern und große Mengen an virtuellem Speicher nutzen, können durch die Verwendung großer Seiten eine Leistungsverbesserung erzielen.

Anmerkungen:

1. Detaillierte Informationen zur Ausführung des Befehls **vmtune** oder **vmo** finden Sie in den AIX-Handbüchern.
2. Sie müssen äußerst vorsichtig vorgehen, wenn Sie Ihr System für das Reservieren von Speicher und die Unterstützung großer Seiten konfigurieren. Wenn Sie zu viel Speicher reservieren, führt das zu aufwendigen Seitenwechselvorgängen für Hauptspeicherseiten, die nicht im Speicher zu belassen sind. Wenn Sie zu viel physischen Speicher für große Seiten zuordnen, wird die Systemleistung beeinträchtigt, wenn nicht ausreichend Speicher für die Unterstützung der 4 KB-Seiten vorhanden ist.
3. Das Festlegen der Registrierdatenbankvariablen DB2_LGPAGE_BP impliziert ebenfalls, dass der Speicher reserviert ist.

Voraussetzungen:

Sie arbeiten in einer AIX 5.x oder höheren 64-Bit-Umgebung. Sie müssen über Root-Berechtigung verfügen, um mit AIX-Betriebssystembefehlen zu arbeiten.

Vorgehensweise:

Gehen Sie wie folgt vor, um die Unterstützung für große Seiten zu aktivieren:

1. Konfigurieren Sie Ihren AIX-Server für die Unterstützung großer Seiten:

   Für AIX 5.1-Betriebssysteme: Setzen Sie den Befehl **vmtune** mit den folgenden Markierungen ab:

   ```
   vmtune -g <GroßeSeitengröße> -L <GroßeSeiten>
   ```

   Für AIX 5.2-Betriebssysteme: Setzen Sie den Befehl **vmo** mit den folgenden Markierungen ab:

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

   Dabei gilt Folgendes:

   `<GroßeSeitengröße>`

   Gibt die Größe in Byte der hardware-unterstützten großen Seiten an.

   `<GroßeSeiten>`

   Gibt die Anzahl der zu reservierenden großen Seiten an.

Wenn Sie zum Beispiel 25 GB für die Unterstützung großer Seiten zuordnen müssen, führen Sie den Befehl folgendermaßen aus:

Für AIX 5.1-Betriebssysteme:

```
vmtune -g 16777216 -L 1600
```

Unter AIX 5.2-Betriebssystemen:

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

2. Führen Sie den Befehl **bosboot** so aus, dass die zuvor ausgeführten Befehle **vmtune** oder **vmo** beim nächsten Booten wirksam werden.
3. Nachdem der Server gestartet wurde, aktivieren Sie ihn für reservierten Speicher:

   Für AIX 5.1-Betriebssysteme: Setzen Sie den Befehl **vmtune** mit den folgenden Markierungen ab:

   ```
   vmtune -S 1
   ```

   Für AIX 5.2-Betriebssysteme: Setzen Sie den Befehl **vmo** mit den folgenden Markierungen ab:

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

4. Setzen Sie mit Hilfe des Befehls **db2set** die Registrierdatenbankvariable DB2_LGPAGE_BP auf den Wert „YES" und starten Sie anschließend DB2 UDB:

   ```
   db2set DB2_LGPAGE_BP=YES
   db2start
   ```

Zugehörige Konzepte:

- „Aufbau von Tabellenbereichen" auf Seite 100
- „Vom Betriebssystem verwalteter Speicherbereich (SMS)" auf Seite 104
- „Von der Datenbank verwalteter Speicherbereich (DMS)" auf Seite 106

Anhang D. 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 322

Zugehörige Tasks:

- „Aufrufen der Kontexthilfe über ein DB2-Tool” auf Seite 341
- „Aktualisieren der auf Ihrem Computer oder Intranet-Server installierten Komponente 'DB2 Information - Unterstützung'” auf Seite 333
- „Aufrufen der Hilfe für Nachrichten über den Befehlszeilenprozessor” auf Seite 343
- „Aufrufen der Hilfe für Befehle über den Befehlszeilenprozessor” auf Seite 343
- „Aufrufen der Hilfe für den SQL-Status über den Befehlszeilenprozessor” auf Seite 344

Zugehörige Referenzen:

- „DB2-Dokumentation in PDF-Format und gedrucktem Format” auf Seite 334

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 324

Zugehörige Tasks:

- „Aktualisieren der auf Ihrem Computer oder Intranet-Server installierten Komponente 'DB2 Information - Unterstützung'" auf Seite 333
- „Anzeigen von Themen in der gewünschten Sprache in 'DB2 Information - Unterstützung'" auf Seite 334
- „Aufrufen von 'DB2 Information - Unterstützung'" auf Seite 331
- „Installation von 'DB2 Information - Unterstützung' mit dem DB2-Installationsassistenten (UNIX)" auf Seite 326
- „Installation von 'DB2 Information - Unterstützung' mit dem DB2-Installationsassistenten (Windows)" auf Seite 329

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 322

Zugehörige Tasks:

- „Aktualisieren der auf Ihrem Computer oder Intranet-Server installierten Komponente 'DB2 Information - Unterstützung'" auf Seite 333
- „Installation von 'DB2 Information - Unterstützung' mit dem DB2-Installationsassistenten (UNIX)" auf Seite 326
- „Installation von 'DB2 Information - Unterstützung' mit dem DB2-Installationsassistenten (Windows)" auf Seite 329

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 322
- „DB2 Information - Unterstützung: Installationsszenarios" auf Seite 324

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 333
- „Anzeigen von Themen in der gewünschten Sprache in 'DB2 Information - Unterstützung'" auf Seite 334
- „Aufrufen von 'DB2 Information - Unterstützung'" auf Seite 331
- „Installation von 'DB2 Information - Unterstützung' mit dem DB2-Installationsassistenten (Windows)" auf Seite 329

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 322
- „DB2 Information - Unterstützung: Installationsszenarios" auf Seite 324

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 333
- „Anzeigen von Themen in der gewünschten Sprache in 'DB2 Information - Unterstützung'" auf Seite 334
- „Aufrufen von 'DB2 Information - Unterstützung'" auf Seite 331
- „Installation von 'DB2 Information - Unterstützung' mit dem DB2-Installationsassistenten (UNIX)" auf Seite 326

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 322

Zugehörige Tasks:

- „Anzeigen von Themen in der gewünschten Sprache in 'DB2 Information - Unterstützung'" auf Seite 334
- „Aufrufen der Kontexthilfe über ein DB2-Tool" auf Seite 341
- „Aktualisieren der auf Ihrem Computer oder Intranet-Server installierten Komponente 'DB2 Information - Unterstützung'" auf Seite 333
- „Aufrufen der Hilfe für Nachrichten über den Befehlszeilenprozessor" auf Seite 343
- „Aufrufen der Hilfe für Befehle über den Befehlszeilenprozessor" auf Seite 343
- „Aufrufen der Hilfe für den SQL-Status über den Befehlszeilenprozessor" auf Seite 344

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 324

Zugehörige Tasks:

- „Aufrufen von 'DB2 Information - Unterstützung'" auf Seite 331
- „Installation von 'DB2 Information - Unterstützung' mit dem DB2-Installationsassistenten (UNIX)" auf Seite 326
- „Installation von 'DB2 Information - Unterstützung' mit dem DB2-Installationsassistenten (Windows)" auf Seite 329

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 322

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 99. 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 100. 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 100. 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 101. 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 101. 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 102. 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 103. 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 104. 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 105. 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 106. 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 106. 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 107. 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 321

Zugehörige Tasks:

- „Drucken von DB2-Büchern mit PDF-Dateien" auf Seite 340
- „Bestellen gedruckter DB2-Bücher" auf Seite 340
- „Aufrufen der Kontexthilfe über ein DB2-Tool" auf Seite 341

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 322

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 340
- „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 334

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 340

Zugehörige Referenzen:

- „DB2-Dokumentation in PDF-Format und gedrucktem Format" auf Seite 334

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 331
- „Aufrufen der Hilfe für Nachrichten über den Befehlszeilenprozessor" auf Seite 343
- „Aufrufen der Hilfe für Befehle über den Befehlszeilenprozessor" auf Seite 343
- „Aufrufen der Hilfe für den SQL-Status über den Befehlszeilenprozessor" auf Seite 344
- „Verwenden der DB2 UDB-Hilfe: 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 341
- „Aufrufen von 'DB2 Information - Unterstützung'" auf Seite 331
- „Aufrufen der Hilfe für Nachrichten über den Befehlszeilenprozessor" auf Seite 343
- „Aufrufen der Hilfe für den SQL-Status über den Befehlszeilenprozessor" auf Seite 344

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 331
- „Aufrufen der Hilfe für Nachrichten über den Befehlszeilenprozessor" auf Seite 343
- „Aufrufen der Hilfe für Befehle über den Befehlszeilenprozessor" auf Seite 343

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 322
- „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".
- Sie können die Größe und Farbe der verwendeten Schriftarten in den DB2-Schnittstellen anpassen. Weitere Informationen hierzu finden Sie unter „Eingabehilfen für Bildschirme".
- DB2-Produkte unterstützen Anwendungen mit Eingabehilfen, die mit der Java™ Accessibility API arbeiten. Weitere Informationen hierzu finden Sie unter „Kompatibilität mit Unterstützungseinrichtungen" auf Seite 347.
- Die DB2-Dokumentation steht in behindertengerechtem Format zur Verfügung. Weitere Informationen hierzu finden Sie unter „Dokumentation im behindertengerechten Format" auf Seite 347.

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 347

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 Syntax-

elementen 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 346

Zugehörige Tasks:

- „Inhaltsverzeichnis"

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 E. 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

E

F

G

H

I

K

L

M

N

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

SC12-3057-01